A modern Approach To Machine Learning In Insurance Fraud Detection

By Gunnar Windsand Sem

15th of August 2019

Supervisor Dr Alexandru V. Asimit

A dissertation submitted as part of the requirements for the award of MSc in Business Analytics

Abstract

Sabre Insurance has introduced a few innovative policies for drivers. However, far reaching and popular these policies are, it’s important to prevent their misuse. Insurance fraud happens to be one of the major concerns in the insurance industry, since it’s not trivial to detect fraudulent claims. Machine Learning can be helpful in detecting frauds in real time, scaling to the dynamically growing data and performing accurately enough. This research aims to report the creation of a bunch of classification models on an anonymized dataset. After removing redundant variables from the data, one-hot encoding transforms categorical features into a numerical 0/1 form, while *k*-Nearest Neighbour and Random Forest help impute missing values. Several bagged or boosted decision tree varieties were trained on a major part of the data, and subsequent models were used to predict fraud instances from the remaining data. Accuracy metrics like Mean Square Error and Area Under the ROC Curve were used to evaluate performance of the models, where 87% of the claims could be correctly identified by an ensemble of ten different models.

Lastly, the research aims to explain these models in such a manner that people with a general business background would be able to understand the theorem and apllication behind the various models. Shapley Additive explanations provided post-model interpretations regarding importance of features, and how each individual feature influenced the probability that a claim could be fraud. It is expected that this kind of high-level classification and introspection into the causes of insurance frauds with Machine Learning would not help Sabre stay safe and secure but provide a glaring example for the insurance industry to follow suit.

Acknowledgment

I want to take this opportunity to thank all my fellow peers that have helped me learn so much about diversity this year, the spectrum of nationalities at Cass Business School has had a significant impact on my academical progression and personal development.

Furthermore, I would give a huge thanks to my supervisor Dr Alexandru V. Asimit, for excellent support on this research, but more importantly for being a course director continuously showing his interest and support for each and every student at the Business Analytics course throughout this master.

I would also like to thank Dr Rui Zhu, Dr Rosalba Radice and Alan Chalk for always being available to my team and guide us through this research process.

Finally, thank you to all my family and friends that continuously show their support and push me to thrive in whatever I'm doing.

Table of Contents

List of tables

List of figures

Company Presentation

Sabre is one of the most successful insurance companies in the United Kingdom, and it has been selling to an extensive network of brokers since 1982. It offers customers and brokers excellent customer service. Further, it is regulated and authorized by the Financial Conduct Authority and Prudential Regulation Authority *(FRN 202795) (Sabre Insurance, 2017).* The company has managed to achieve critical milestones in history since its inception. One of the most recent milestones accomplished in 2017 is the company's trade on the London Stock Exchange. Sabre has come up with multiple distribution strategies by selling its policies through three direct brands. The brands are Go Girl, Insure 2 Drive, and Drive Smart (GO GIRL, insure2drive and **DRIVE SMART**).

The first brand is Go Girl, which is an insurance for women. Although it is easy to find insurance covers for women, insurance companies have very few policies specifically catered to female drivers. Sabre has gone a notch higher by providing insurance covers for women. The Go Girl insurance cover offers services such as handbag cover, uninsured driver promise, personal accident cover, free legal cover, and a free courtesy car (*Sabre Insurance, 2015*). This insurance cover can be customized to the price that suits the customer.

The other brand is Insure 2 Drive, which is the trading name for Sabre Insurance Company. This brand was launched in November 2010 and offered internet-only private car insurance product. The last one is the Drive Smart car insurance policy, which assists drivers to get a better deal. The company uses the latest technology to quantify safety in driving. The parameters that are measured include length of trips, extent of nighttime driving, braking, congestion, speeding, and driver's familiarity with the road (*Sabre Insurance, 2019*). Drive Smart uses a telematics gadget which sends data about the speed and distance of trips. Additionally, the smart drive box acts as a free theft tracker. The drivers can use the tracker to show how safely they are driving.

Introduction - Why Machine Learning in Insurance Fraud Detection

Fraud management has been a significant challenge in the commerce and banking industries. The insurance industry has not also been left behind with the number of fraud cases increasing each day. Therefore, machine learning is one of the strategies that can be adopted to eliminate fraud.

The insurance industries consist of thousands of companies worldwide. The companies such as Sabre Insurance company collect more than one trillion dollars each year. When an entity or a person makes false insurance claims to attain insurance benefits or compensation, they are referred to as insurance fraud. A research carried out showed that the total cost of insurance fraud could be projected to more than forty billion dollars (*Roy & George, 2017*). Therefore, the detection of insurance fraud has been a significant challenge in the insurance industry that is currently experiencing decline in income (*2019 Insurance Industry Outlook, Deloitte).*

When building the detection models, the savings from loss deterrence have to be well-adjusted with the cost of false alerts. Machine learning is one of the techniques which has been applied to advance predictive accuracy and enable the loss control unit to attain higher coverage with low false positive rated. Insurance fraud covers a broad spectrum of activities which individuals commit to attaining a satisfactory outcome from the insurance company (*Wipro Limited, 2019*). The potential situations are twisting the context of the incident, expanding the impact of the incident, and coming up with a situation that was not covered under the insurance. The companies managing to utilize this process experience a fall in necessary processing time and cost, and a good increase in service quality (Towards Data Sciene, 2019)

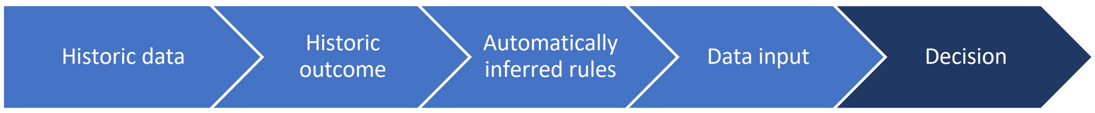


Figure 1: Machine learning process in insurance

The traditional tactic for fraud detection is founded on developing heuristics around the fraud indicators. A decision on fraud is made based on the heuristics (*Wipro Limited, 2019*). In specific occasions, rules are framed which defined to ascertain whether the case needs to be investigated. Further, a checklist can be developed, showing the scored for the various indicators of fraud. An aggregate of these scores with the value of the claim determines if the case requires further investigation (*Wipro Limited, 2019*). The criteria for determining the thresholds and indicators are tested periodically and statistically recalibrated.

However, using this method has several limitations. One is forced to operate with a restricted set of known parameters based on the heuristic knowledge while being aware that other qualities would influence decision making. The other limitation is the inability to the specific relationship between different parameters such as insurance sale process, customer segment, and geography (*Wipro Limited, 2019*). Different industry experts state that there is no typical model; hence, there is a significant challenge in determining the model in a specific context. Moreover, recalibration is a manual exercise, hence the ability to carry out calibration is a challenging process.

From the challenges above, it is evident that using traditional statistics perspective is a significant challenge. Therefore, insurance companies have decided to leverage machine learning capability. Machine learning aims to present a variety of data to the algorithm without judgment on the significance of data elements (*Wipro Limited, 2019*). The intent for machine learning is to develop a model that can be tested on the known frauds through diverse algorithmic methods. The aim of using machine learning techniques is to expand the accuracy of recognition.

In this research, the data was split into three categories, which are testing and training. The exercise was conducted on transaction data for the company, which applied a combination of pre-defined rules and predictive machine learning algorithms to identify the outliers in data. The big data platform was made up of three layers, which include data handling, detection layer, and outcomes. Under data handling, there was data cleansing, tokenizing, and transformation. Under the detection layer, there was ML Algorithms and business rules. Lastly, on outcomes, there are case management, detailed reports/tables, and visualizations. The datasets were classified into numerical and categorical data which has been anonymized.

However, there are many challenges to this method of fraud detection. One of the major challenges is encountered in the process of machine learning handling of both categorical and missing values. One strategy for handling missing value is multiple imputations. The missing imputation procedure replaces each of the missing value with a set of plausible values where in this research, there was used two specific machine learning techniques— *k*-Nearest Neighbour (*k*-NN) and Random Forest (RF) in the following procedure:

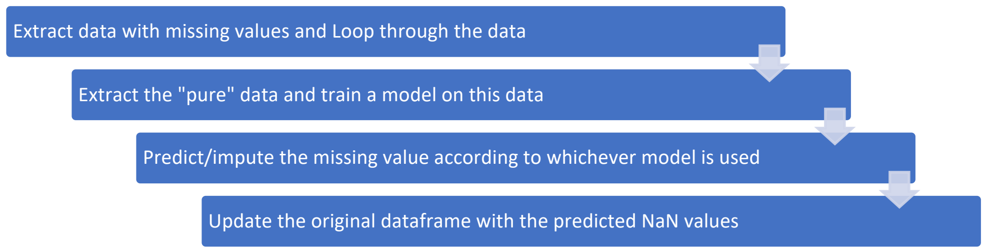

Extract data with missing values and Loop through the data
Extract the "pure" data and train a model on this data
Predict/impute the missing value according to whichever model is used
Update the original dataframe with the predicted NaN values

Figure 2: Missising values imputation

This process is resulting in predicted accuracies with a mean of 92% for the imputations of missing values(Figure 10 in appendix)

Although there are different challenges experienced in machine learning, the success of the machine learning algorithms rests on how the data is represented. Machine learning has been a useful strategy which helps individuals to recognize most of the fraudulent cases with a low false-positive rate that is based on reasonable precision (*Wipro Limited, 2019*). Feature engineering is one of the processes that can be applied to alter raw data into features that better signify the fundamental problem. For example, functions could be used to remove intercorrelated variables that would create "noise" when applying the different machine learning techniques and reducing precision.

The set of models provide a practical suit to apply in insurance claim fraud. Nevertheless, the models have to be custom-made based on particular business context and user priorities.

Overly, three factors explain why insurance companies should used machine learning in fraud detection. The first one is the speed. Machine learning can evaluate many transactions in real-time. It continuously analyses new data.
The other important factor is scale. Machine learning is effective, with increased data sets (*Maruti Techlabs, 2019*). Machine learning improves more with data as it can pick out the similarities and differences between multiple behaviours.
The last factor is efficiency as machine learning can perform repetitive tasks. Therefore, it is more effective than humans at detecting non-intuitive or subtle patterns to help identify fraudulent transactions.

Supervised Learning

Supervised learning is the most widely-used category of tasks in practical machine learning. It is a process where a host of input variables are utilized to infer decision about an output variable. Generally an algorithm is utilized to learn how the output can be predicted from the input variables (*Brownlee, 2016*). The aim is to estimate the mapping function representing this interrelation so that one can have an input data to forecast the output. It is referred to as supervised learning because the training dataset represents a teacher supervising the learning process. The learning process stops when the algorithm attains an adequate level of performance.

Additionally, supervised learning is a type of learning where the systems follow a pre-determined pattern. It can be described as learning akin to what is practised by human beings, *viz.* gaining understanding from past experiences to obtain knowledge in accomplishing real-world tasks. With supervised learning, there is an application of algorithms to produce a reasonable prediction of response from the new data (*Murphy, 2019*).

The supervised learning process offers a distinct tool to categorize and process data using the machine learning language. Supervised learning uses branded data, which is data classified to infer a learning algorithm. The data set is used as a basis to predict the classification of the unlabeled data through the machine learning algorithm. It is the most common category of machine learning algorithms, which is used in medical imaging research (*Murphy, 2019*). Supervised learning is different from unsupervised learning, as, in unsupervised learning, the outputs are unknown. It is divided into two categories, which are classification and linear regression.

Primarily, supervised learning begins with training data which are tagged with the correct answers in this case target values. It is a method which is used to classify objects, situations and problems based on the associated data which is fed into the machines. The machines are fed with data such as dimensions, patterns, and characteristics repetitively until the machine can perform accurate classification (*Shobha & Rangaswamy, 2018*). Lastly, supervised learning is a method that is used to segment customers based on customer data and offer product recommendations. Therefore, Sabre insurance company can use this technique to segment their customers based on the customer data available.

Theoretical background

Several classification and regression methods were partaken in the model. The principal machine learning paradigm used for predicting frauds was boosted decision trees. Several varieties of trees were used, namely gradient boost machines, extreme gradient boost and distributed random forest. A regression tree was also used along with the *k*-nearest neighbour to predict the missing values and fill in the gaps in the dataset. The following subsections deal with the mathematical background of the methods and algorithms on which different models were built, and the statistical measures used to evaluate the models.

While the major algorithms used in the analysis and modelling (*viz.* *k*-Nearest Neighbour, Random Forest, two Boosting methods and an Ensemble learning combining these different algorithms) are discussed in the first five subsections, the error and performance metrics used to evaluate the models' performance are discussed in the next three subsections. Mean Square Error (MSE), Confusion Matrix and Area Under the Receiver Operating Characteristic Curve (AUC-ROC) has been used to evaluate the performance of different classifiers.

k-Nearest Neighbour (k-NN)

One of the simplest and popular algorithms in machine learning is the *k*-Nearest Neighbour (*Tenkomo, 2006*). The method relies on finding a few close matches rather than constructing a general model to fit to the entire data, and is hence a typical example of *lazy learning*. Just as learned from experience and develop a kind of heuristic reasoning that prompt the user to act in a way much like observed others doing successfully, *k*-NN is based on a similar kind of instance-based learning. An unknown object or observation is classified as the majority class to which the *k* nearest objects in the training set belong.

There are different measures to calculate distance or nearness, of which the Euclidean distance is the most acceptable and commonly used norm:

$$d(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

where $X \equiv (x_1, x_2, \ldots, x_n)$ and $Y \equiv (y_1, y_2, \ldots, y_n) \in \Re^n$ are two points in the Cartesian space of dimension n representing the observation to classify and a similar instance to follow. Smaller the distance $d(X, Y)$ more the similarity. In this way the k most similar instances are selected in a pool, and class of the unknown observation is predicted to be the majority class from the instance pool.

The method works well for most practical purposes where the data size is small to moderate, but prediction time builds exponentially as the number of instances grow. Another limitation of this method is that it is not possible to assign different weights to different samples. Optimal performance of *k*-NN also heavily depends on the choice of *k*, which needs an appropriate pilot study. The *k*-NN method has been used in the current analysis to impute missing values, whereas the actual classification is done by the three algorithms that follow.

Random Forest (RF)

Another machine learning technique with a simple structure yielding interpretable results is the *decision tree* (DT). While DTs can predict classes of objects (like whether a transaction could be fraud or is possibly safe) with a good level of confidence in most cases, a few data pockets get ignored due to the generalizing nature of a single tree. This is because a DT always searches for general patterns in data, finding which it develops a *branch* to encompass that generalized behaviour in the form of a rigid rule. Flexing the branch or tweaking such a rule is something that is next to impossible in the domain of a single tree.

Random forest resolves this lack of flexibility by growing a bunch of trees and making actual predictions by consulting and comparing the individual predictions from each of the trees in a forest. This technique is generally referred to as *bagging*. In most cases the final prediction goes by the majority of the individual predictions. Since trees trying to generalize a large dataset take up a good bit of time, it is customary to use a *sample* of the dataset to build simpler trees comprising the random forest.

Gradient Boosting Machines (GBM)

Instead of building a bunch of trees simultaneously, the gradient boosting machine (*Friedman, J. H. 2001*) constructs trees iteratively, learning from previous trees how to minimize the prediction error effectively. Bagging (as in RF) and *boosting* (as in GBM) can be viewed as two different approaches to create ensembles of weaker decision trees, where GBM is a step-wise learner expected to yield better and better trees as the learning progresses iteratively. It has a closed-loop system where the feedback is induced along the direction of the steepest decent over the error hyperplane to modify and improve the performance of trees by optimally cutting down the error rate.

GBM essentially improves on a model $M_k(X)$ by fitting an additive model to the residuals

$$g(X) = y - M_k(X)$$

and constructing the next model as:

$$M_{k+1}(X) = M_k(X) + \hat{g}(X).$$

where $\hat{g}(X)$ is the predictor model fit on $g(X)$ in iteration $k+1$.

EXtreme Gradient Boosting (XGBoost)

Extreme gradient boosting is a modern variation of the Gradient Boosting concept described above, which utilizes the parallel processing capability of cloud computing, and exploits any sparsity in the data. It is in fact an efficient and scalable implementation of the general GBM framework proposed by Friedman (*Friedman, J. H. 2001*). What makes XGBoost unique and highly popular, is the more regularized model formalization that effectively controls over-fitting, rendering it a superior performance compared to other DT-based models.

Apart from its parallelizable capacity and superior performance, XGBoost has been noted to be 10 times faster than GBM (*Chen, T. & He, T. 2019*). It can therefore be considered to be the perfect choice to achieve accurate predictions for large datasets in a big data framework.

Stacked Ensemble (SE)

In machine learning, *ensemble* is an agglomeration of a host of different learning algorithms, often quite dissimilar and complimentary to each other. The objective is to obtain more accurate predictions than could be achieved from any one algorithm predicting individually. In the strict sense, Random Forests and Gradient Boosting Machines are also ensemble learners. They constitute a set of similar homomorphous weak learners (*i.e.* decision trees) and form a single, strong learner. On the other hand an ensemble of RF, GBM and XGBoost would in fact be an ensemble of several strong learners with different capabilities. While implementing an RF is like taking vote among workers in a store on what kind of modification would improve production the most, constructing a Stacked Ensemble is like taking suggestions from a committee of experts from different traits (like finance, production, management and marketing) to form a decision on the topic.

H2O's Stacked Ensemble is a supervised ensemble machine learning functionality that employs the optimal combination of prediction algorithms through what is called stacking. Also known as super-learning, it is an algorithm that trains a second-level metalearner to find and implement a combination of the constituent base learners whose weighted vote will give the most accurate predictions in most of the test cases. It was proved (*van der Laan, M., Polley, E. & Hubbard, A. 2007*) in 2007 that the Super Learner ensemble gives an optimal performance in any machine learning task.

Mean Square Error (MSE)

While it is the objective of the prediction to learn how to predict the value of a target variable as closely and accurately as possible, practical machine learning algorithms are never perfect. An error, however large or small, exists in every prediction, which is given by

$$\begin{aligned}\varepsilon &= \text{Actual value} - \text{Predicted value}\\ &= y - \hat{y}\end{aligned}$$

As the predicted value could be both greater or less than the actual value of any particular observation, summing up the errors over all observations can lead to cancellation of positive and negative values. It is thus customary in Statistics to calculate the squares of errors and sum up to give the sum of squares of errors. When divided by the number of observations, this gives the mean square error (MSE)— an important indicator of the perfection achieved in prediction:

$$\text{MSE} = \sum_{i=1}^{n} \varepsilon_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Evidently, lower the MSE, better is the predictor.

Confusion Matrix

Mean square error unfortunately suffers from quite a few limitations, like dependence on scale of the target value, high sensitivity to outliers, and incapability to handle class imbalance. On the other hand, a better view of the quality of prediction could be obtained by looking at the distribution of predictions of all instances belonging to each class. Such a distribution would provide a compact summary of the output containing all necessary information, provided not to distinguishing between instances of the same class. It can be illustrated in the form of a matrix called the confusion matrix. For a binary class variable taking up just two values (as in the present insurance fraud problem), this would look like:

Table 1. Confusion matrix for a binary class problem

		Actual Class	
		0	1
Predicted Class	0	TN	FP
	1	FN	TP

Considering all observations with a target value "1" as the positive class (denoting the fraud claims), they can either be truly predicted as "1" or falsely predicted as "0"— leading to some true positive (TP) and some false positive (FP) instances. Similarly all negative observations (actual class "0") can either be predicted as true negative (TN) or false negative (FN). Consolidating the number of instances falling into each of these four mutually exclusive categories, resulting in a form of a confusion matrix illustrated in Table 1.

Area Under the ROC Curve (AUC-ROC)

Sometimes, instead of making direct predictions as "0" and "1", the probabilities of belonging to either of the classes may be calculated in a prediction algorithm. This leaves the user with the discretion of deciding with how certainty one wants to predict each class— often determined by the misclassification cost of the classes. A curve can be obtained by plotting the true positive rate (TPR) against the false positive rate (FPR) at various thresholds or cut-off probabilities beyond which an instance will be classified into that particular class.

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \qquad \mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}}$$

This curve is known as the Receiver Operating Characteristic (ROC) curve. A typical ROC curve is presented in Figure 3.

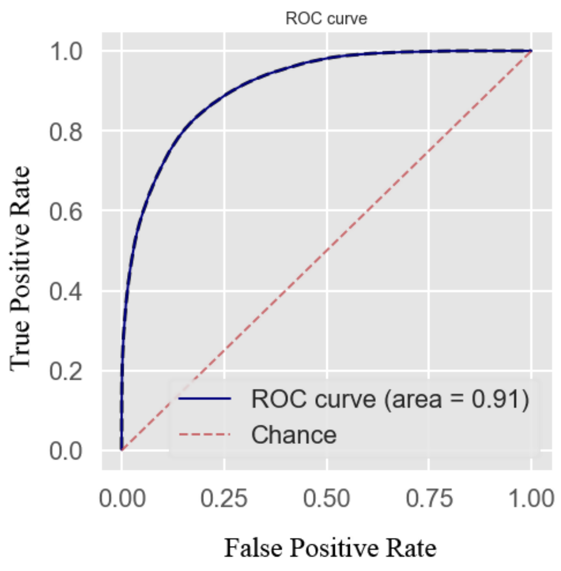


Figure 3. An ROC curve is essentially the FPR vs. TPR plot at different thresholds

The Area Under this Curve (AUC-ROC) is accepted by most data scientists as a robust measure of performance of a classifier. A wild guess instead of an appropriate machine learning model would lead to an ROC curve depicted in Figure 3 as the red dotted line (labelled "Chance"), which would give an AUC-ROC of 0.5. Thus AUC of any classifier is typically more than 0.5 and an AUC close to 1.0 indicates a nearly perfect classification.

Whereas ROC may be viewed as a probability curve, AUC represents the degree or measure of separability. It tells how much a model is capable of distinguishing between classes (like "fraud" and "not fraud"). Higher the AUC, better the model is at predicting "0"s as "0"s and "1"s as "1"s. Another way of interpreting AUC-ROC is the probability of a model ranking a random positive example higher than a random negative example (*Google Developers, 2019*).

Methodology

To study the implication of machine learning in predicting incidents of insurance fraud, this research has been carried out using different approaches, techniques and models applying the Python programming language for in-depth analysis. In the following subsections the necessary python packages that need to be imported (along with their specific use in the different modelling paradigms) are listed in step 1, data cleaning and manipulation methods are discussed in step 2, while steps 3 and 4 deal with the details of model testing and post-model interpretation.

Step 1: Data Cleaning/manipulation

The dataset contains 179567 rows (or observations) and 89 columns (or features), of which "Var47" is the target or output variable. The features were annonymized *a priori* to preserve confidentiality of sensitive insurance data and protect privacy of the customers. The objective of this research is to **predict the binary outcome "Var47" from the other features.**

"Var47" was renamed "target" to keep it semantically discernible from other annonymized features. The distribution of the target variable was quite balanced, with nearly equal percentage of observations in the non-fraud (0) and fraud (1) categories. There are also a few (less than 10%) values missing. A summary of this class distribution is presented in Table 2.

Table 2. Distribution of the binary "target" variable (along with count of missing values)

Target value	No. of observations	Percentage of observations
0	83855	47%
1	81174	45%
missing	14538	8%
Total	179567	100%

Redundancy

Two input features, *viz.* “uniqueID” and “train” were removed from the feature set considered for modelling, as they were unrelated with the classification model. The former assumes different values for each observation and serves as a unique identifier for each customer interaction, whereas the latter is a binary column indicating whether that particular observation should be included in the training set or not.

Besides these two, a good degree of dependence was present among the other 86 input variables as well. To assess the degree of dependence, the *conditional entropy* (*Cover, T. M., & Thomas, J. A. 1991*) between all input variable pairs were calculated as:

$$\mathrm{H}\left(Y|X\right) = -\sum_{\substack{x\in \mathrm{X} \\ y\in \mathrm{Y}}} \wp\left(x,y\right) \log \frac{\wp\left(x,y\right)}{\wp\left(x\right)}$$

which gives a measure of the information needed to completely determine a random variable Y given the value of another random variable X, where $\wp(x, y)$ denotes the probability that $X = x$ and $Y = y$, $\wp(x)$ denotes the probability that $X = x$, and ⇪, ⇪ are support sets of X and Y. Only one variable from each pair with $\mathrm{H}\,(Y \mid X) > 0.9$ was taken, while the other was dropped from the set of features. In this process 10 more variables were removed.

A heatmap of inter-variable correlations further showed that “Var54” was highly (94%) correlated with “Var24”. The former was hence dropped, and the number of input features subsequently reduced to 75.

Encoding categorical variables

Some of the features were categorical and some numerical. There were 39 categorical variables, which do not commensurate with building models in Python libraries— whether it be *k*-NN, RF or boosted trees. These variables were hence encoded as binary ‘dummy’ variables, by creating new columns for every value of each of the categorical variables— also known as ‘one-hot encoding’ (*Potdar, K., Pardawala, T. S., & Pai, C. D. 2017*). This resulted in a dataset with 136 input features, all of which had numeric values (some floating point real numbers, some integers, and a good majority of them— particularly the one-hot encoded variables— with just binary values 0 or 1). A typical example of one-hot encoding “Var_11” is shown in Table 3.

Table 3. Variable columns before and after one-hot encoding for just one feature "Var_11"

Var_11		Var11_B	Var11_C	Var11_D	Var11_E	Var11_F	Var11_G
E		0	0	0	1	0	0
B	one-hot $\xrightarrow{}$	1	0	0	0	0	0
B	encoding	1	0	0	0	0	0
D		0	0	1	0	0	0
F		0	0	0	0	1	0

The possible values of the categoric variable "Var_11" are 'B', 'C', 'D', 'E', 'F' and 'G'. This requires creating six different columns for each of the variable-values. The respective values under each column will be '1' if the concerned variable takes up that particular value, and '0' otherwise.

Handling missing values

Missing values were present in 18 columns of the transformed data. Of these, columns with more than 30% of their values missing were removed. Missing values in the other columns were imputed through prediction from *k* Nearest Neighbour (*k*-NN) or Random Forest (RF)— whichever gave higher accuracy of fit. In this way around 0.5% of the values in the dataset were imputed. However a threshold of 6 was taken to distinguish between categoric and numeric values. Variables with less than 6 distinct values were predicted through classification, whereas those variables with 6 or more distinct values were predicted through regression. The resulting predicted values were imputed into the missing value positions, and a 'clean' dataset was obtained containing no missing values.

Splitting into Training and Test datasets

The whole dataset was split into training and test sets. The objective is to use the training dataset to build the model, and predict the accuracy of the model by predicting the output ("target") from the input features in the test set. If the model predicted the target feature of all the test set observations well, concluding that the model is well-built and reliable.

The training set comprised of 70% of all observations, whereas the rest 30% was assigned to the test set for blind prediction.

Step 2: Model building and testing

The 136 features in the training dataset was normalized to values in the [0, 1] interval using the `StandardScaler` function from `sklearn.preprocessing` library, so that all features get same weights in the model. Two different models were run on the standalone computer, while a big data interface was used to run a host of other models including a stacked ensemble.

On standalone computer

First the `LightGBM` model was fit to the training data. The parameter settings were as follows:

- `metric = "auc"`
- `application = "binary"`
- `learning_rate = 0.5`
- `max_depth = 10`
- `num_leaves = 80`
- `verbosity = -1`
- `verbose_eval = 150`
- `num_boost_round = 1000`
- `early_stopping_rounds = 100`

The `CatBoostClassifier` was the second model available from the `catboost` library. Parameter here were set to the following values:

- `eval_metric = "AUC"`
- `iterations = 4000`
- `depth = 10`
- `od_type = "Iter"`
- `loss_function = "CrossEntropy"`
- `logging_level = "Silent"`

A maximum tree depth of 10 was set in both cases, which gives sufficiently large trees with adequate predictive ability. AUC was also set to be the evaluation metric in both cases. The `LightGBM` was set to construct sparse trees with a maximum of 80 leaves, and 1000 boosting rounds, while the `CatBoostClassifier` was allowed to run for a maximum of 4000 iterations with the `CrossEntropy` as the measure of check for branching.

On the big data framework

Finally, the python module H2O was used to build a few models in the big data framework. H2O is a Java-based software for data modeling which is available as a python module providing access to the H2O Java Virtual Machine *(The H2O Python Module).* The module provides privileges for distributed computing with many machines, parallel processing with many CPUs, and a powerful engine with several hundred GBs in memory allocation. H2O is oriented around simple horizontal scaling with the "divide and conquer and combine" paradigm along with a concurrent application structure to get faster solutions.

In the big data framework the `H2OGeneralizedLinearEstimator` was used for a preliminary GLM fitting function before more complex models were fit. While the resulting AUC-ROC (0.84) was somewhat lower than other model performances, it served as a base accuracy and pilot model to test the integrity of the framework. Subsequently Distributed Random Forest (DRF), Gradient Boosting Machine (GBM) and eXtreme Gradient Boost (XGBoost) models were used on a Stacked Ensemble learner with adaptive coefficients that were dynamically determined by a metalearner. The results are presented in the Results section.

The models built were evaluated from the prediction accuracy on the test dataset. Different metrics discussed in the Assumptions section were used to determine how well each model was able to predict some of the fraud cases that were not included in the model training, but specifically kept aside for evaluating the model's predictability. While MSE is a domain-specific measure, stating in absolute terms the average error made by the model in predicting any random insurance claim, the confusion matrix and AUC-ROC gave more in-depth view into what kind of things went wrong, and where the model performed really well.

Step 3: Model interpretability

Machine learning models serve a two-fold purpose. One of them comprises the primary goal of predicting 'which' insurance claims are possibly fraudulent. The extent to which this objective is served can be measured by several metrics like MSE or AUC-ROC.

However, understanding 'why' a model predicts a claim to be fraud could be no less crucial than knowing the prediction itself. Unfortunately, most models are too complicated or have inter-connected layers (*e.g.* the Stacked Ensemble learner) to get a clear logical answer to 'why' a certain claim has been classified as fraud. From the classical view, a certain transparency or interpretability of the model may help to get a satisfactory answer to this question. Unfortunately this classical view limits the algorithms that can be used to a handful few, and is often bound to trade-off with accuracy in prediction (which happens to be the primary goal). This is even more true in the big data era, where most models resemble a 'black-box' in the sense that the user gets very little idea of what is going on inside the model. Nonetheless, with state-of-the-art techniques *(Lundberg, S.M. and Lee, S-I. 2017),* it is possible to separately investigate the effect of a variable on the model from several aspects to know its individual influence on the predictions. Some variables may have higher significance, while some may be relatively insignificant in distinguishing the frauds. Another aspect, independent of the magnitude of significance, is the positive or negative influence on the model. When one variable increases, the probability of fraud may rise, whereas other variables may have a negative influence on the chances of fraud. A third insight is given by looking deeper into the respective influences on higher or lower variable values on the output. For example a low-budget claim is less likely to be fraud— a kind of interpretation that reveals very important information about customers and helps keeping transactions secure.

In the present study three kinds of interpretations are illustrated as a post-processing stage of the model. The first encompasses a bar chart representing the relative influence of the 10 most important variables, where the variable with the highest significance is given an importance of 1.0. Most of the machine learning models in python has this functionality built inside the classifiers as an added functionality. The second and third model interpretations involve the *direction* and *range of values* across which each variable influences the predicted class. For that a modern feature analytics approach (discussed below) is taken refuge of, that unifies a few existing methods with some new techniques to convey the interpretations in an intuitive way.

Shapley Additive explanations (SHAP) is a unified approach *(Lundberg, S. M., Nair, B., Vavilala, M. S., Horibe, M., Eisses, M. J., Adams, T., ... & Lee, S. I. 2018)* to interpret in depth the interrelations on which a machine learning model is built. SHAP utilizes tenets of game theory and some local explanations, uniting them with several previous methods to produce consistent feature attribution modules based on expectations or probability theory. The parallel and complementary nature of SHAP with any prediction or classification model is sketchily illustrated in Figure 4.

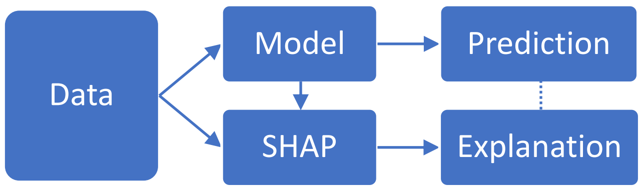


Figure 4. Model interpretability with SHAP

A force-plot for a single observation can be obtained with the `shap.force_plot()` function from the `shap` python library. It illustrates the strength by which each variable forces the risk of fraud in case of a particular observation. For example, in case of observation # 92 (see Figure 7) the risk of fraud increases from −0.1602 (its base or average value over all observations) to 0.51 (the predicted risk of observation # 92). Variables in red push the risk higher, whereas variables in blue push the risk lower. The magnitude of force exerted by each variable on this particular observation is given by the length of the (red or blue) strip it occupies in the figure.

Force plots for the entire series of observations, rotated by 90° and arranged horizontally give a stacked force plot (see Figure 8). This provides a summary of the influence of the major variables in case of all observations. The plot used in the present report contains all observations arranged according to similarity in the *x*-axis, which makes interpretations easier through similarity-based clusters, rather than when arranged arbitrarily.

A `shap.summary_plot` indicates the manner and extent to which lower or higher values of different important variables impact the output. For example, Figure 9 shows point clutters along each variable has both red and blue dots. The higher values of a variable are marked in red and lower values of a variable in blue. Red dots cluttered on the right side indicate that higher values of that variable increase the risk of fraud, whereas blue dots jumbled up near the *x*-axis origin would mean that lower values hardly influence this risk.

Results

Ten models were built with the `AutoML` function from the `h20.automl` library. The models include—

- 1 Distributed Random Forest (DRF) model,
- 3 eXtreme Gradient Boost (XGBoost) models,
- 4 Gradient Boosting Machines (GBM), and
- 2 Stacked Ensemble (SE) classifiers.

The `AutoML` Stacked Ensembles use a Generalized Linear Model (GLM) with non-negative weights as its default metalearner algorithm (see the relevant discussion section). The area under the Receiver Operating Characteristic curve (AUC-ROC) and Mean Square Error (MSE) were calculated as the measures of performance of these models (see discussion). The values obtained are plotted in Table 4.

Table 4. Performance of the 10 models built to predict insurance fraud

Model Name	**AUC-ROC**	**MSE**
DRF_1	0.862	0.153687
XGBoost_1	0.867	0.149278
XGBoost_2	0.867	0.150501
XGBoost_3	0.849	0.161641
GBM_1	0.864	0.150267
GBM_2	0.866	0.149176
GBM_3	0.867	0.148510
GBM_4	0.869	0.147698
SE_BestOfFamily	0.870	0.147770
SE_AllModels	0.870	0.147760

From the classification accuracies calculated from AUC-ROC, the two ensemble models exhibit the best performance, followed by the GBM_4 model. The coefficient of participation of different models in the Standard Ensembles can be visualized in Figure 5

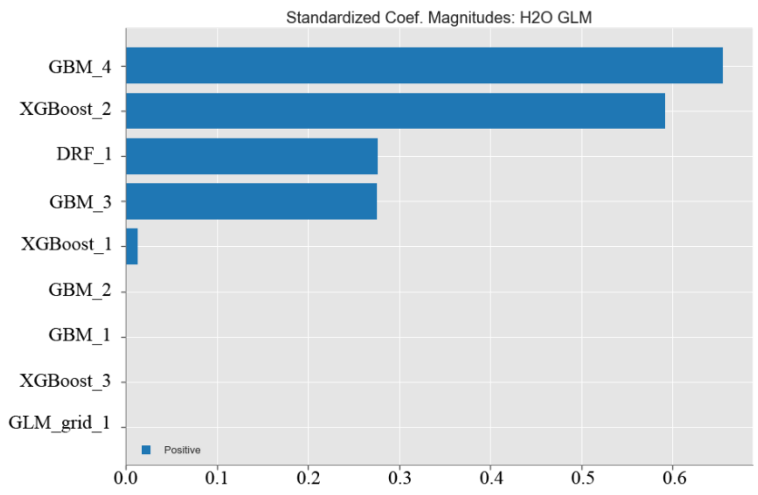


Figure 5. Participation of different models in the Standard Ensembles

Clearly, from the first eight models in Table 4, those which show better performance (*viz.* GBM_4, GBM_3, XGBoost_2 and XGBoost_1) take part in the ensembles with higher coefficient magnitudes— thus enabling the ensemble predictor to perform at its best.

The test set consisted of 165029 observations, of which 83855 were "0" and 81174 were "1". The predictions for each of the classes can be best visualized in the form of a confusion matrix presented in Table 5.

Table 5. Confusion Matrix for classification with the GBM_4 model

		Predicted Class		
		0	**1**	**Error**
Actual Class	0	64078	19777	0.236
	1	10006	71168	0.123
	Total	74084	90945	0.181

Once classification is done, the significance of variables can be judged using the `h2o.varimp_plot` method. Variable importance is determined by calculating the relative influence of each variable, *i.e.* whether a particular variable was selected for splitting during the tree building process, and how much the squared error (over all trees) decreased as a result of considering that variable.

The following bar chart in Figure 6 shows the relative importance of the 10 most significant variables using the XGBoost and GBM processes, calculating the importance as a fraction of the most important variable found.

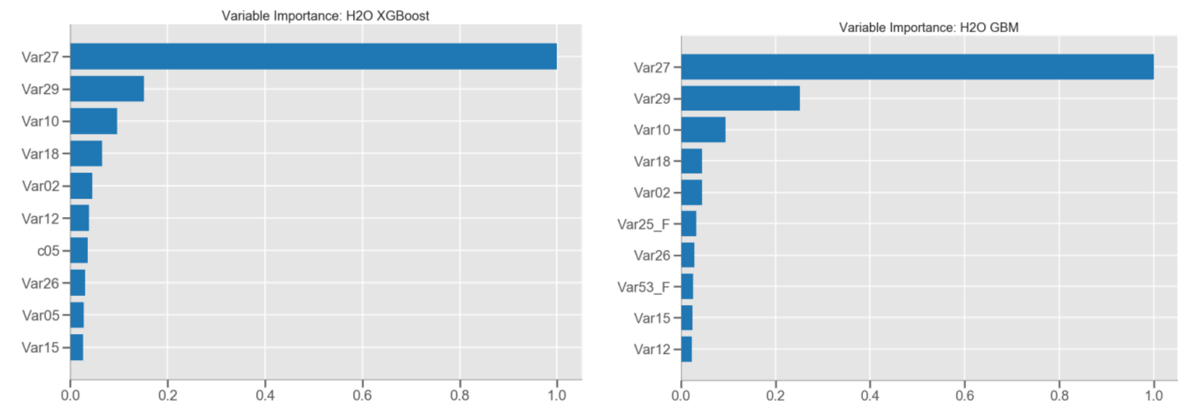


Figure 6. Relative importance of variables selected during building trees in the XGB and GBM processes

Both the XGB and GBM classifiers indicate that "Var27" is by far the most significant variable— nearly 5 times more important than any other variable. "Var29", "Var10" and "Var18" emerge to be the next significant features. The anonymized nature of the data precludes any further conclusion to be drawn from this visualization. However, given the real significance of these variables, further knowledge could be gained from this post-model feature ranking analysis.

The impact of each variable on a particular observation (# 92) can be visualized through a force-plot presented in figure 7, depicting variables that push the risk of fraud higher in red and variables that pull down the risk of fraud (*i.e.* the safety-inducing variables) in blue. Further, the length of the strips of each variable in the red-blue bar is an indicator of how strongly the feature increases or reduces the risk.

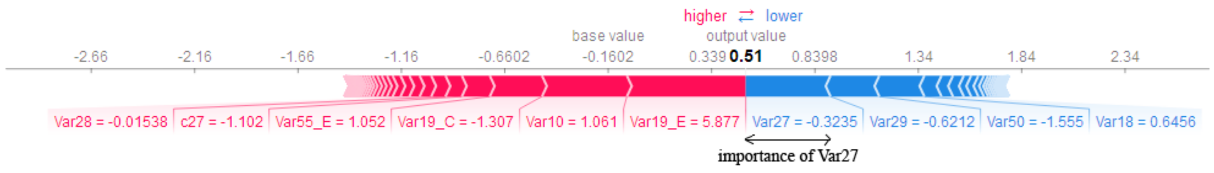


Figure 7. Force plot showing (positive or negative) impact of variables on observation # 92

For example, "Var19_E" occupying the longest part of the bar and located on the red-side, may be interpreted to have the highest influence on the 92nd observation in the direction of *increasing* the fraud risk by a magnitude of 5.87. On the other hand, "Var27" the longest blue strip, may be interpreted to *decrease* the risk by a relative magnitude of −0.32. The resultant pushing and pulling of the different variables have increased the risk of observation # 92 from −0.1602, the base (or average) risk of all insurance claims in the dataset, to 0.51.

Now, this kind of force plot can be obtained for each of the hundred thousand observations, but it is barely possible to interpret them individually. Plotting the rotated force-plots of all observations side-by-side however gives a grand picture of how each variable influences the output on an overall basis. This plot is shown in figure 8.

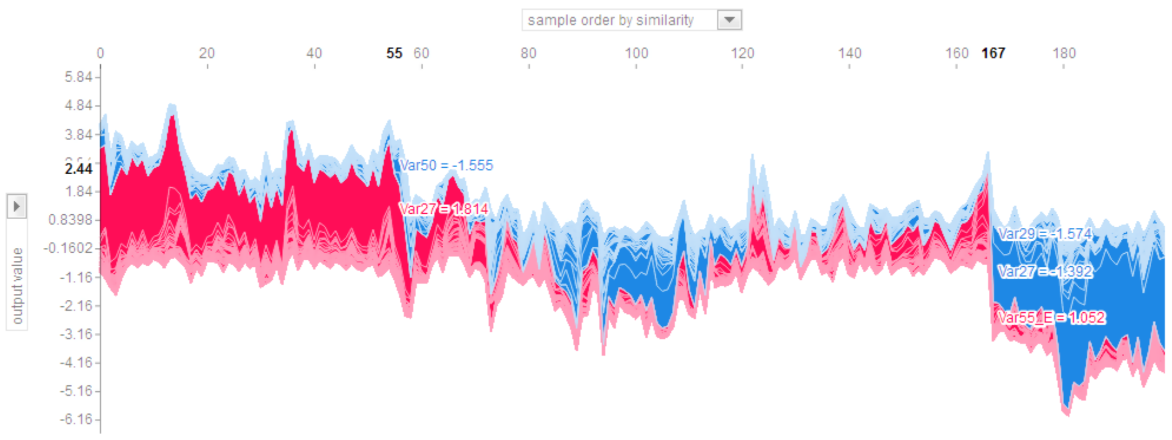


Figure 8. Force plot of all observations stacked side-by-side showing influence of all input variables on the output variable

This in an interactive plot in the Jupyter Notebook created by python, and one can choose to view the influence of any one (or all) of the variable values on the output, or even on any other input variable. In this respect the stacked force-plot is an immense visualization possibility. The above figure 8 shows the influence of all variables on the output (*i.e.* risk of a particular insurance claim being fraud) of observations arranged according to similarity. In particular bringing the mouse pointer over any selected observation would give the numeric values of the major responsible variables that helped predict that observation. Values for observation 55 and 167 are illustrated, which may be interpreted as: (i) "Var50" decreases (and "Var27" increases) the risk of fraud for observation 55 by a magnitude of −1.555 (1.814), whereas (ii) "Var29" and "Var27" decrease (and "Var55_E", one of the dummy variables, simultaneously increases) the risk of observation 167 being fraud by a magnitude of −1.574 and −1.392 (1.052) respectively.

Finally, SHAP is used to generate a plot illustrating which range of values of the more significant variables are responsible for raising or lowering the risk of insurance fraud.

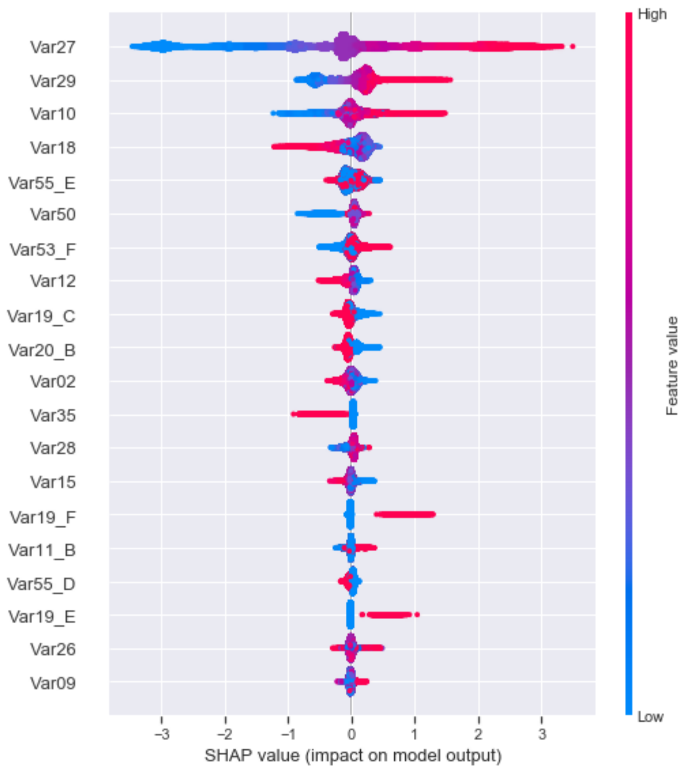


Figure 9. Influence of different value ranges of the more significant variables on the output

The interpretations of the plot run like this: from the red portions of the horizontal clutters along each variable, it can be inferred that higher values of "Var27", "Var29" or "Var10" has a positive impact on the model output (*i.e.* increases the risk of fraud), but higher values of "Var18" has a negative impact (*i.e.* lowers the fraud risk). On the other hand, lower values of "Var27" and "Var10" lowers the risk more than "Var29", but nothing much can be strongly inferred about the chance of fraud from a lower value of "Var18".

Conclusion

The results of this research prove that utilizing modern machine learning techniques will suffice as a powerful analytical tool in insurance fraud detection. By applying machine learning in insurance fraud detection, you see examples of companies that reach significant percentages in ROI, which could be critical for companies like Sabre Insurance in an industry that experiences a decline in income. Furthermore, companies who have been able to adopt automation of some aspects of their fraud claim process can experience a significant fall in processing time and cost, and a definite increase in service quality.

Quoting Towards Data Science article about the "black box" metaphor in machine learning;

"Most, if not all of these deep learning approaches, or even more generally machine learning approaches are, essentially black boxes, in which you can't really inspect how the algorithm is accomplishing what it is accomplishing."

This research now contradicts this statement from 2017 proving that applying modern interpretability techniques in machine learning are allowing companies such as Sabre Insurance to understand which factors and variables are driving the models, and why they are producing the specific output.

As machine learning/AI is continually evolving, it's only down to people's imagination trying to foresee how far this will have come in a few years.

"The pace of progress in artificial intelligence is incredibly fast. Unless you have direct exposure to groups like Deepmind, you have no idea how fast—it is growing at a pace close to exponential."

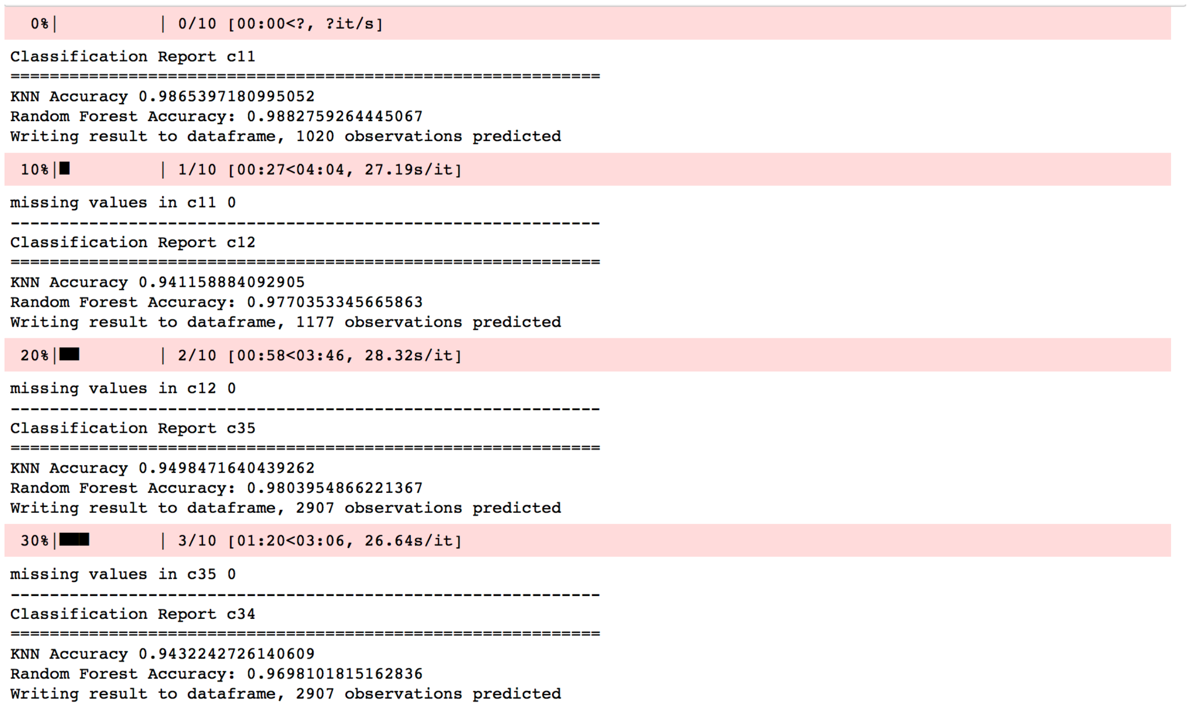
-- Elon Musk

Appendix

Table 6. Packages and Functions imported in the python script

No.	Package name	Function name	Utility
1.	csv		read csv files
2.	datetime		provide date/time
3.	re		string matching
4.	warnings		syntax warnings
5.	math		math functions
6.	sys		system functions
7.	pandas		main math library
8.	pandas_profiling		calculate runtimes
9.	numpy		numerical library
10.	matplotlib.pyplot		plot graphs charts
11.	seaborn		statistical graphic
12.	scipy.stats		statistical method
13.	catboost		GBM, XGBoost
14.	gc		memory cleaning
15.	shap		explain outputs
16.	os		operating system
17.	mpl_toolkits.mplot3d	Axes3D	draw graph axes
18.	collections	Counter	loop interface
19.	tqdm	tqdm	display progress bar
20.	sklearn.ensemble	IsolationForest	find DT splits
21.	sklearn.ensemble	RandomForestClassifier	RF classification
22.	sklearn.ensemble	RandomForestRegressor	RF regression
23.	sklearn.neighbors	KNeighborsRegressor	*k*-NN classification
24.	sklearn.neighbors	KNeighborsClassifier	*k*-NN regression
25.	sklearn.preprocessing	OneHotEncoder	one-hot encoding
26.	sklearn.preprocessing	LabelEncoder	re-encode labels
27.	sklearn.preprocessing	StandardScaler	scaling features
28.	sklearn.preprocessing	Normalizer	normalize features
29.	sklearn.decomposition	PCA	does PCA with SVD
30.	sklearn.model_selection	train_test_split	split data for train/test
31.	sklearn	neighbors	build *k*-NN model
32.	sklearn	metrics	accuracy metrics
33.	numpy.random	permutation	randomize searches
34.	keras.models	Sequential	layered models
35.	keras.layers	Dense	keras layer type
36.	catboost	CatBoostClassifier	train models
37.	catboost	Pool	data processing
38.	catboost.utils	get_roc_curve	draw ROC curve
39.	catboost.utils	get_fpr_curve	draw FPR curve
40.	catboost.utils	get_fnr_curve	draw FNR curve
41.	h2o.estimators.glm	H2OGeneralizedLinearEstimator	build GLM model
42.	h2o.estimators.gbm	H2OGradientBoostingEstimator	build GBM model
43.	h2o.estimators.random_forest	H2ORandomForestEstimator	build RF model
44.	h2o.estimators.deeplearning	H2ODeepLearningEstimator	deep learning model
45.	h2o.grid.grid_search	H2OGridSearch	optimize ensembles
46.	h2o.automl	H2OAutoML	build ensemble model

Figure 10: Results from imputation method



```
  0%|          | 0/10 [00:00<?, ?it/s]
Classification Report c11
============================================================
KNN Accuracy 0.9865397180995052
Random Forest Accuracy: 0.9882759264445067
Writing result to dataframe, 1020 observations predicted
 10%|█         | 1/10 [00:27<04:04, 27.19s/it]
missing values in c11 0
------------------------------------------------------------
Classification Report c12
============================================================
KNN Accuracy 0.941158884092905
Random Forest Accuracy: 0.9770353345665863
Writing result to dataframe, 1177 observations predicted
 20%|██        | 2/10 [00:58<03:46, 28.32s/it]
missing values in c12 0
------------------------------------------------------------
Classification Report c35
============================================================
KNN Accuracy 0.9498471640439262
Random Forest Accuracy: 0.9803954866221367
Writing result to dataframe, 2907 observations predicted
 30%|███       | 3/10 [01:20<03:06, 26.64s/it]
missing values in c35 0
------------------------------------------------------------
Classification Report c34
============================================================
KNN Accuracy 0.9432242726140609
Random Forest Accuracy: 0.9698101815162836
Writing result to dataframe, 2907 observations predicted
```

References

Brownlee, J., 2016. *Supervised and Unsupervised Machine Learning Algorithms*. [Online] Available at: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/
[Accessed 20 July 2019].

Maruti Techlabs, 2019. *How Machine Learning Facilitate Fraud Detection?* [Online] Available at: https://www.marutitech.com/machine-learning-fraud-detection/
[Accessed 21 July 2019].

Murphy, A., 2019. *Supervised learning (machine learning*). [Online] Available at: https://radiopaedia.org/articles/supervised-learning-machine-learning
[Accessed 20 July 2019].

Roy, R. & George, K. T., 2017. Detecting insurance claims fraud using machine learning techniques. *IEEE Xplore*, 1(1), pp. 32 - 37.

Sabre Insurance, 2015. *Car Insurance for Women*. [Online] Available at: https://gogirl.co.uk/
[Accessed 20 July 2019].

Sabre Insurance, 2017. *About Us*. [Online] Available at: https://sabre.co.uk/about/
[Accessed 20 July 2019].

Sabre Insurance, 2019. *Drive Smart Explained*. [Online] Available at: https://drivesmartinsurance.co.uk/#about-us
[Accessed 20 July 2019].

Shobha, G. & Rangaswamy, S., 2018. Computational Analysis and Understanding of Natural Languages: Principles, Methods and Applications. *Handbook of Statistics*, 1(1), pp. 1 - 14.

Wipro Limited, 2019. *Comparative Analysis of Machine Learning Techniques for Detecting Insurance Claims Fraud*. [Online]
Available at: https://www.wipro.com/analytics/comparative-analysis-of-machine-learning-techniques-for-detectin/
[Accessed 21 July 2019].

Google Developers. (2019). Classification: ROC Curve and AUC | Machine Learning Crash Course | Google Developers. [online]
Available at: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc [Accessed 23 Jul. 2019].

Teknomo, K. (2019). *K Nearest Neighbors Tutorial*. [online] People.revoledu.com. Available at: https://people.revoledu.com/kardi/tutorial/KNN/ [Accessed 24 Jul. 2019].

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. Annals of Statistics, pages 1189–1232.

Chen, T. & He, T. (2019). Xgboost: extreme gradient boosting. R package version 0.90.0.1, 1-4.
van der Laan, M., Polley, E. & Hubbard, A. (2007). Super Learner. Statistical Applications in Genetics and Molecular Biology, 6(1), Article 25.

Cover, T. M., & Thomas, J. A. (1991). Elements of Information Theory John Wiley & Sons. New York, 68, 69-73.

Medium. (2019). *How AI Enables Smarter Claims Processing & Fraud Detection?*. [online] Available at: https://towardsdatascience.com/how-ai-enables-smarter-claims-processing-fraud-detection-e65a8b2997a6 [Accessed 25 Jul. 2019].

Potdar, K., Pardawala, T. S., & Pai, C. D. (2017). A comparative study of categorical variable encoding techniques for neural network classifiers. International Journal of Computer Applications, 175(4), 7-9.

The H2O Python Module http://docs.h2o.ai/h2o/latest-stable/h2o- py/docs/intro.html

Lundberg, S.M. and Lee, S-I. (2017). A Unified Approach to Interpreting Model Predictions. In 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Lundberg, S. M., Nair, B., Vavilala, M. S., Horibe, M., Eisses, M. J., Adams, T., ... & Lee, S. I. (2018). Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. Nature biomedical engineering, 2(10), 749.

Card, D. (2017). The “black box” metaphor in machine learning. [online] Medium. Available at: https://towardsdatascience.com/the-black-box-metaphor-in-machine-learning-4e57a3a1d2b0 [Accessed 21 Jul. 2019]

FRISS. (2018). *Anadolu Sigorta Customer Story FRISS Fraud Detection*. [online] Available at: https://www.friss.com/customer-story/anadolu-sigorta/ [Accessed 1 Aug. 2019].

Deloitte. (2019). *2019 Insurance Industry Outlook*. [online] Available at: https://www2.deloitte.com/global/en/pages/financial-services/articles/gx-insurance-industry-outlook.html [Accessed 2 Aug. 2019].

Lee, R. (2014). *Elon Musk Fears Artificial Intelligence? 'Something Seriously Dangerous' May Happen in 5 Years*. [online] Tech Times. Available at: https://www.techtimes.com/articles/20521/20141120/elon-musks-nightmare-about-artificial-intelligence-something-seriously-dangerous-may-happen-in-the-five-year-timeframe.htm [Accessed 5 Aug. 2019].