IT1244 Project Report: Use of Machine Learning for Predicting Fraud in Electricity and Gas Consumption

Team 27

Kevin Khoo Weixue
Nguyen Thanh Tam
Tan Wenyuan, Ignatius

Introduction

Fraud in electricity and gas consumption presents a concern for companies, consumers and society. This problem is particularly acute in developing countries, with losses amounting to more than 5% of total energy generated in Brazil and Mexico. (Flick and Morehouse 2010). Fraudulent consumption results in financial losses that are not only borne by distribution companies, but are also passed onto consumers in most cases. Additionally, illicit tampering with distribution networks compromises the reliability of the power grid and poses a risk to public safety (Plummer 2025).

Due to the large amount of data associated with electrical and gas consumption, machine learning is often used to predict and identify fraudulent activity.

The techniques employed by past works done on fraud detection include:

- k-Nearest Neighbours and Decision Trees; however, these algorithms are over-sensitive to noise and class imbalance in the data, with poor recall of 0.05 and 0.2 respectively (Gresoi, Stamatescu and Făgărășan 2025).
- Extreme Gradient Boosting (XGB), which achieved high recall of 0.9 but a concerning false positive rate of 0.03 (Buzau et al. 2018)
- Natural Gradient Boosting (NGB); the model had a high recall of 0.91, at the expense of a high false positive rate of 0.048 (Hussain et al. 2021).

The limitations of the aforementioned algorithms can mostly be explained by a few unique characteristics of the task of fraud prevention, as detailed below:

- Highly imbalanced dataset, as fraudulent consumption forms only a small minority of all transactions.
- Enforcement requires in-person field inspections and so is costly, especially for developing countries with limited financial resources (Gresoi et al. 2025). Hence it may be more important to have a low false positive rate than high accuracy, if enforcement costs pose a limiting constraint on the number of enforcement visits that can be carried out.
- Computational costs should be kept low, as money saved from lower computational costs could be spent on more enforcement visits that would increase real-world effectiveness (Oprea and Bâra 2022).
- The decision-making algorithm should be interpretable, in order to aid regulatory reporting and build trust with corporate stakeholders (Coma-Puig et al. 2016).

It can be observed from the literature that it is challenging to find a good balance between recall, false positive rate, computational costs and interpretability when predicting fraud. Considering this, our project aims to investigate the effectiveness of various machine learning algorithms.

Dataset

The dataset consists of two files, “client.csv” and “invoice.csv”. The former contains 5 features of identifying information for around 11,000 clients, with each client labelled according to their involvement in fraud or lack thereof. The latter comprises the transaction records of these clients, containing 12 features inclusive of consumption levels, invoice date and tariff type. The dataset does not contain any null values, and thus there was no need for any null value handling of the data.

We observed a few issues with the datasets. The distribution of clients is highly imbalanced, with only 4.4% engaged in fraudulent behaviour. This is consistent with the nature of electrical and gas consumption data, as seen from the literature. Additionally, several features in the datasets contain nominal categorical data that is represented by a

string or an integer. This presents a problem even after the feature has been converted into discrete numerical data and scaled, as the algorithm may assume that the assignment of different numbers to each category corresponds to an order or hierarchy between categories. Hence, we obtain binary data using **one-hot encoding** to alleviate concerns of algorithmic bias. Lastly, as each client has multiple transaction records in the invoice dataset, we must aggregate each client's transactions before merging both datasets. Taking these concerns into consideration, we have decided to pre-process the data as follows:

For the client dataset:

- Convert 'dis' (district), 'catg' (category) and 'region' from integer to binary data by one-hot encoding.
- Use 'date' in conjunction with the invoice dataset to compute the duration between the client joining and the client's most recent transaction.

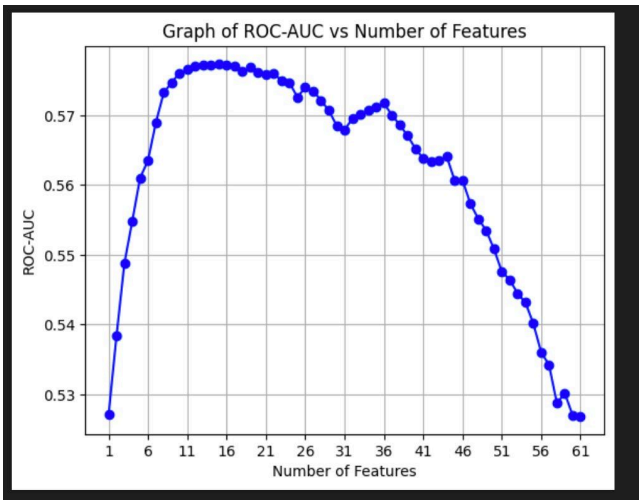
For the invoice dataset:

- Convert 'counter_type' from string to binary data by one-hot encoding.
- Convert 'date' from string to a pandas datetime object, then aggregate 'date' by computing the median, maximum and minimum duration between successive transactions of each client.
- Aggregate 'consommation_level_1', 'consommation_level_2', 'consommation_level_3', 'consommation_level_4', 'counter_coefficient' and 'reading_remarque' for each client by computing the mean, maximum, minimum and standard deviation of each feature.
- Feature scaling is conducted by standardising all aggregated numerical features, in order to prevent variations in magnitudes from affecting the weightage given to each feature.
- Finally, we use the common feature 'id' to merge both datasets, resulting in a total of 60 features.

Methods

We first used the **train-test split** method to ensure that the models will not be trained on any of the testing data, so as to avoid overfitting. This is done by randomly stratifying the dataset by 'id' to divide the dataset into 80% training data and 20% testing data.

Due to the large number of features that we have extracted from the dataset, it is necessary to perform feature selection in order to reduce the computational load of the classification algorithm and reduce overfitting. We used 'SequentialFeatureSelector' from the module 'feature_selection' in 'scikit-learn'. By using **Sequential Forward Selection** (SFS), the transformer iteratively adds the feature with the best cross-validation score until the desired number of features is selected. We plot the ROC-AUC Score against the number of features, and determine the appropriate number of features to be 11, based on the elbow method as shown below.



From this process, the 11 selected features are as given: 'counter_type_ELEC', 'counter_type_GAZ', 'region_103', 'region_302', 'region_304', 'region_305', 'region_308', 'region_311', 'region_371', 'consommation_level_1_min', 'consommation_level_2_std'.

Subsequently, we use the **Synthetic Minority Oversampling Technique** (SMOTE) to address class imbalance in the dataset. SMOTE over-samples the minority class, i.e. fraudulent clients, by synthetically generating new data points. The algorithm takes a minority sample, chooses one of its k-nearest neighbours and randomly creates a new synthetic example that is between the two points, repeating until the minority class is equal in size to the majority class. We implement SMOTE using 'SMOTEENN' from 'combine' in the library 'imblearn'. This resampling method also uses Edited Nearest Neighbours (ENN) to clean the data by removing noisy or borderline samples that have a different class label from a majority of its k-nearest neighbours.

We then evaluate the dataset using three different classification algorithms, all of which utilise supervised machine learning.

Logistic Regression (LR)

Logistic Regression represents the two classes of 'fraud' and 'not fraud' with a binary value of 0 or 1. LR then uses the sigmoid function $g(z) = \frac{1}{1+e^{-z}}$ to predict the probability that a given sample belongs to a class, returning a probability value from 0 to 1. The model predicts the class of each sample based on whether $g(z)$ is greater than or smaller than 0.5. During training, the LR model adjusts its parameters to minimise the sum of the errors between the predicted probability and the actual binary value of

each sample. For our implementation, we use the 'scikit-learn' library to import 'LogisticRegression' from the module 'linear_model'.

LR has a relatively low computational cost and is highly interpretable, as the predicted probability of each sample and the features taken into consideration can be easily identified. However, LR is less effective for non-linear relationships, which reduces its recall.

Support Vector Machine (SVM)

Support Vector Machine finds the optimal hyperplane to form a decision boundary separating data points of different classes. The choice of the hyperplane ensures that the margin is maximised, by widening the distance between the hyperplane and the nearest data points, which are also known as support vectors. Unlike LR, SVM does not predict probability estimates but rather decision values. During training, the SVM model adjusts its parameters to further minimise classification error. We use the 'scikit-learn' library to import 'SVC' from the module 'svm'.

Unlike LR, SVM is able to model non-linear decision boundaries by making use of kernel functions. In addition, SVM also handles high-dimensional data well, which is relevant due to the large number of features involved in fraud prediction. Nevertheless, SVM is computationally expensive, especially for large datasets, and is also hard to interpret as the relative importance of the features used cannot be easily identified (Jindal et al. 2016).

Gradient Boosting

Gradient Boosting builds an ensemble of decision trees that are individually weak learners. The model starts with a single regression tree, following which the sum of the errors between the tree's predictions and their actual class values is computed based on the error function. Thereafter, the algorithm adds new trees iteratively. Each new tree predicts the errors from the previous model (as opposed to the final class output), and the new predictions are multiplied by a small learning rate before it is added to the previous model's predictions.

We implement the technique using **Extreme Gradient Boosting** (XGB), which takes advantage of parallel processing and regularisation to offer improved performance. The algorithm is imported as 'XGBClassifier' from the 'xgboost' library.

Like SVM, XGB handles non-linear relationships well and is highly scalable, allowing it to process large datasets efficiently. XGB is also more interpretable than SVM, as feature importance can be evaluated through their gain, weight and cover, albeit it is still less interpretable than LR (Tsao, Y., Rahmalia, D. and Lu, J. 2024).

Hyperparameter Tuning

After running our three vanilla models on the test dataset to compare the performance of each default model on the test dataset, we then run hyperparameter tuning in the form of **RandomizedSearchCV**, which helps to select an optimal set of parameters through a search on randomized combinations from specified distributions (*RandomizedSearchCV,* n.d.). This is performed over the more common **GridSearchCV** in the interest of efficiency, as it enables us to tune many different parameters without taking too long a time.

Results & Discussion

We shall first discuss the target recall for the three vanilla models. To recap, in the context of this problem, recall measures the proportion of actual fraudulent clients that were correctly identified by the model. In the context of fraud detection, recall is an important metric, as high recall ensures that most of the fraudulent cases are captured, although at the expense of incorrectly classifying non-fraudulent cases as fraudulent. This tradeoff is acceptable in this case as failure to detect fraudulent cases is a detrimental problem financially, while 'false alarms' can be manually reviewed, which will be much easier as there are much fewer fraudulent clients than non-fraudulent clients in practice, and thus much fewer expected clients classified as fraudulent than non-fraudulent.

Logistic Regression (Vanilla)

Model result on test dataset:

	precision	recall	f1-score	support
0	0.96	0.87	0.91	2246
1	0.06	0.18	0.09	103
accuracy			0.84	2349
macro avg	0.51	0.53	0.50	2349
weighted avg	0.92	0.84	0.88	2349

Support Vector Machine (Vanilla)

Model result on test dataset:

	precision	recall	f1-score	support
0	0.96	0.84	0.90	2246
1	0.06	0.23	0.10	103
accuracy			0.82	2349
macro avg	0.51	0.54	0.50	2349
weighted avg	0.92	0.82	0.86	2349

Extreme Gradient Boosting (Vanilla)

Model result on test dataset:

	precision	recall	f1-score	support
0	0.96	0.85	0.90	2246
1	0.04	0.13	0.06	103
accuracy			0.82	2349
macro avg	0.50	0.49	0.48	2349
weighted avg	0.91	0.82	0.86	2349

As we can see from the results, we can see that although the precision and recall on the non-fraud cases are relatively good with a consistent F1-score in the 0.9 region, the target precision and recall, and thus the F1-score suffers quite a lot in comparison. Although we did perform SMOTE to address the class imbalance while training, the synthetic samples may not be able to fully capture the complexity and patterns of actual fraudulent cases, and thus perform relatively poorly on the test dataset. However, out of the 3 models, the Support Vector Machine has a target recall of 0.23, which is the highest out of the three.

After performing hyperparameter tuning using RandomizedSearchCV, where we explored a wide range of relevant hyperparameters with F1-score as our scoring metric, the results were as follows.

Logistic Regression (Hyperparameter-tuned)

Model result on test dataset:

	precision	recall	f1-score	support
0	0.96	0.60	0.73	2246
1	0.05	0.43	0.08	103
accuracy			0.59	2349
macro avg	0.50	0.51	0.41	2349
weighted avg	0.92	0.59	0.71	2349

Support Vector Machine (Hyperparameter-tuned)

Model result on test dataset:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	2246
1	0.04	1.00	0.08	103
accuracy			0.04	2349
macro avg	0.02	0.50	0.04	2349
weighted avg	0.00	0.04	0.00	2349

Extreme Gradient Boosting (Hyperparameter-tuned)

Model result on test dataset:

	precision	recall	f1-score	support
0	0.96	0.84	0.90	2246
1	0.05	0.17	0.08	103
accuracy			0.82	2349
macro avg	0.50	0.51	0.49	2349
weighted avg	0.92	0.82	0.86	2349

From the results of the hyperparameter tuning, we can observe that in general, without compromising default precision as much with the exception of the SVM, the target recall has improved across all three models. Although the tuned SVM model achieved a perfect target recall of 1, which suggests that all fraudulent clients were identified, it is important to note that the precision and recall for the default cases were reduced to 0. This suggests that all clients were predicted as fraudulent, and thus rendering the model no better than a naive classifier, and not a sensible model to use. Comparing between the rest of the 5 models, the tuned Logistic Regression model seems to perform the best among them for fraud detection, with a target recall of 0.43.

Although the target recall of the model is relatively low, in the sense that the average human could most likely achieve a better recall rate than this, we believe that this model could work in tandem with human efforts - whereby the model is first used as a preliminary measure to narrow the amount of client transactions that the human has to scan through, and weed out the clients that are actually non-fraudulent among those reported as fraudulent by the model.

References

Flick, T.; and Morehouse, J. 2010. *Securing the Smart Grid: Next Generation Power Grid Security.* Elsevier.

Plummer, D. 2025. *Electricity Theft: Economic Burden And Sustainability Risks.* Forbes.

Gresoi, S.; Stamatescu, G.; and Făgărăşan, I. 2025. Advanced Methodology for Fraud Detection in Energy Using Machine Learning Algorithms. Applied Sciences 15(6): 3361. doi.org/10.3390/app15063361.

Buzau, M.; Aguilera, J. T.; Cruz, P.; and Gomez-Exposito, A. 2018. Detection of Non-Technical Losses Using Smart Meter Data and Supervised Learning. IEEE Transactions on Smart Grid. 99(1): 126-140. doi.org/10.1109/TSG.2018.2807925.

Hussain, S.; Mohd, W. M.; Saeed, F.; and Al-rimy, B. 2021. Sensors. 21(24): 8423. doi.org/10.3390/s21248423.

Oprea, S.; and Bâra, A. 2022. Feature Engineering Solution with SQL Analytic Functions in detecting Electricity Frauds. Scientific Reports. 12(1): 3257-3276. doi.org/10.1038/s41598-022-07337-7.

Coma-Puig, B.; Carmona, J.; Gavalda, R.; Alcoverro, S.; and Martin, V. 2016. Fraud Detection in Energy Consumption: A Supervised Approach. Data Science and Advanced Analytics. 19(1): 120-129. doi.org/10.1109/dsaa.2016.19.

Jindal, A.; Dua, A.; Kaur, K.; Singh, M.; Kumar, N.; and Mishra, S. 2016. Decision Tree and SVM-Based Data Analytics for Theft Detection in Smart Grid. IEEE Transactions on Industrial Informatics. 12(3): 1005-1016. doi.org/10.1109/TII.2016.2543145.

Tsao, Y.; Rahmalia, D; and Lu, J. 2024. Machine-learning Techniques for Enhancing Electricity Theft Detection considering Transformer Reliability and Supply Interruptions. Energy Reports. 12(1): 3048-3064. doi.org/10.1016/j.egyr.2024.08.068.

RandomizedSearchCV. (n.d.). *Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV*