BT4012 Group Project Final Report

—

Detecting Electricity Frauds using Machine Learning

GROUP 30

Matriculation Number	Full Name
A0200073B	**Jed Lee Woon Kiat**
A0220682L	**Jiang Yutong**
A0222591J	**Keltonn Lim Jing Feng**
A0222740R	**Passion Goh Wei Ling**
A0232468B	**Xu Zihan**

Google Drive Link (Recommended):
https://drive.google.com/drive/folders/1OmkP8ddVegAsxxQayw94j0oRr_8DOOFl?usp=sharing

Github Link:
https://github.com/JFLKeltonn/BT4012-Sem-1-2022-Group-30.git

FRAUD TABULAR DATA CLASSIFICATION FOR UTILITY CONSUMPTION

INTRODUCTION

In Singapore, we bear witness to multiple waves of fraudulent activities over the past few years. That is especially so since the onset of the pandemic. Phishing scams involving SMS messages from scammers impersonating bank staff in Singapore have increased significantly, from 149 cases in 2020 to 1,021 in 2021. Some of us personally, have ourselves or know people around us that had experienced or been a victim of some form of fraudulent activities in one way or another. From the most recent Luxury Goods Scam to the spate of phishing scams involving OCBC Bank that resulted in more than S$13.7 million lost earlier this year, we cannot emphasise further the importance of fraud prevention and mitigation. Beyond fraud and scams in the financial sector, there are as well vulnerabilities in the existing system that could potentially affect us directly or indirectly. One such example is utility fraud.

In 2016, SP Services customers received calls informing them that their accounts are in arrears or that their electricity meters need to be changed. They are then asked to make payments to a designated account (Chew, 2016). In 2021, one website, known as Voltex, claims that Singaporeans are overpaying for electricity by $27.6 billion dollars a year, and that “lawmakers” and the “Public Utility Commission” are not preventing electricity companies from running a “crooked business” of selling overpriced electricity to consumers (GOV.SG, 2021).

As a small, resource-constrained country, Singapore imports almost all its energy needs. It is of utmost importance to treat any form of fraud activity or scam seriously. From a data-driven perspective, we decided to focus on utility fraud as we observed and understood the potential impact such fraudulent activities could have. Hence, we decided to dedicate this project to evaluating the various symptoms and characteristics of fraudulent activities and providing insights and recommendations on identifying potential fraud and reducing its negative impact.

DATA SOURCE AND COLLECTION

Link to data: https://www.kaggle.com/datasets/mrmorj/fraud-detection-in-electricity-and-gas-consumption?select=invoice_train.csv

Detecting fraud related to electricity consumption is usually a difficult challenge as the input datasets are sometimes unreliable due to missing and inconsistent records, faults, misinterpretation of meter reading remarks, status, etc. In this report, we obtained meaningful insights from fraud detection using real datasets of Tunisian electricity consumption metered by conventional meters.

The dataset was sourced online via Kaggle under the title ‘Fraud Detection in Electricity and Gas Consumption’, published by the user Andrii Samoshyn. It details the electrical and gas consumption data trends for different clients of the Tunisian Company of Electricity and Gas (STEG). STEG is a public and non-administrative company; it is responsible for delivering electricity and gas across Tunisia. The company suffered tremendous losses in the order of 200 million Tunisian Dinars due to fraudulent manipulations of metres by consumers.

Using the client’s billing history, we primarily aim to detect and recognize clients involved in fraudulent activities. It is with the intention that our analysis could enhance the company’s revenues and reduce the losses caused by such fraudulent activities.

DATA CHARACTERISTICS

The dataset is described by the following characteristics:

1. Duplicates and missing data. Based on preliminary checks, there also do not seem to be missing or null values. All clients in the dataset are unique, with no duplicate data. There exist 11 duplicates in the invoice dataset, which were dropped.

2. Data Imbalance. Based on an initial count of the data, we found that 5.5% of the data set is marked fraudulent. This means the full data set is likely to be imbalanced. To deal with this imbalance, we will be using repeated sampling of fraud data during our model training to make up for the imbalance and improve model performance.

3. Datasets. The dataset provided consisted of 2 tables, a client table, which describes each unique client, and an invoice table, which details the consumption trends throughout time for each client, as well as information regarding the Gas/Electricity Counter used. The client data set contains a total of 135493 unique clients with 5 categorical variables each and one target column indicating fraud or otherwise. The training invoice data set contains 4476738 unique transactions and 16 categorical variables. In total, the dimensionality of the raw dataset is 21, excluding the target column.

   The dataset also came with a 'test' set, provided by the original dataset owner. However, this dataset did not come with labels. In addition, since many of the fields in this dataset are hidden from us, we could not populate the provided test set with labels through manual labelling, since we do not know what labels like 'district 60' means. As such, we had to discard the data.

4. Client fields. The following are the field descriptions for the client dataset:

Field Name	Data Type	Description
Client_id	String	Unique id for client
District	Integer;	District where the client is
Client_catg	Integer	Category client belongs to
Region	Integer	Area where the client is
Creation_date	Date	Date client joined

Target	Binary, Integer	fraud:1 , not fraud: 0

5. Invoice fields. The following are the field descriptions for the invoice dataset:

Field Name	Data Type	Description (Oprea & Bâra, 2022)
Client_id	String	Unique id for the client
Invoice_date	Date	Date of the invoice
Tarif_type	Integer	Type of tax
Counter_number	Integer	Identifier of each counter or meter that belongs to a client
Counter_statue	String, Integer (Messy)	Takes up to 5 values such as working fine, not working, on hold statue, ect
Counter_code	Integer	An additional identifier of the counter
Reading_remarque	Integer	Notes that the STEG agent takes during his visit to the client (e.g: if the counter is tampered, the investigator gives a remark)
Counter_coefficient	Integer	An additional coefficient to be added when standard consumption is exceeded
Consommation_level_1	Integer	Consumption_level_1
Consommation_level_2	Integer	Consumption_level_2
Consommation_level_3	Integer	Consumption_level_3
Consommation_level_4	Integer	Consumption_level_4
Old_index	Integer	Old index; provide the previous and current consumption values
New_index	Integer	New index; the difference between the new and old indexes provide the consumption that is invoiced
Months_number	Integer	Month number; the number of months between readings
Counter_type	String	Type of counter, electricity or gas

DATA PREPROCESSING

The Data was processed through the following ways:

1. Data Type Standardisation. The `counter_statue` field contains multiple entries of differing data types. This field details the remarks a technician has written about the working state of the gas/electricity counter. These remarks have likely been clustered beforehand by the source data provider as they may contain sensitive product information. As it stands, these clusters contain hidden information we cannot make sense of. This field in particular has differing data types. The clusters that counters were assigned to were stored as integers 0 to 5. But we found 3 general error types in this column.

   The first type of error was a type conversion error. Some of the counter status integers were stored as strings. For example, an integer 0 could be stored as a string '0'. For this error, we simply changed the type back to integer.

   The second type of error involved an english character input instead of a number. For example, 'train_Client_30467' had a `counter_statue` of 'A'. Since 'A' was the first character in the alphabet, and the most common value in `counter_statue` was the first number, `0`, we defaulted all values of 'A' to zero.

   The third type of error was a formatting error, which was complicated to resolve. Some of the fields had numbers greater than 5, such as 'train_Client_53725', who had a ridiculous value of 269375. Upon closer inspection, we realised that for these clients, almost all of their data fields were labelled wrongly. In fact, for these rows, the values for each field was shifted one column to the right.

   We verified this by performing a groupby - count function for each column in the invoice table to find the most common data values in these columns. Then, for each column value in the error row, we checked to see if it belonged to a common value in the previous column. We confirmed that, indeed, the values for these rows were shifted one position to the right from their intended column.

   To resolve this error, we shifted the data values one column to the left to re-establish the correct values. The only exception to this was for one row with a `counter_statue` value of 420, where almost all the values in the row was 0. We dropped that single row of data.

   The column was standardised into integers, renamed to `counter_status`, and subsequently was segregated into 6 separate binary encoded columns during one-hot encoding subsequently.

2. Datetime Processing. We included two additional columns for the invoice dataset, 'month_of_year' for the month an invoice and 'year' for the year the invoice was issued. This could be useful for time series analysis later.

3. Train Test Split. We split the training data to generate a test set for training evaluation. Using Scikit-Learn's train test split method, we removed 10% of the existing training set from the client table for the test set to evaluate the model training, and another 10% for the validation set for final model evaluation. This means that the training set contains 80% of all clients, the test set

has 10% of all clients and the validation set has 10% of all clients.

This split was done after the data type standardisation but before all other pre-processing steps to prevent data leakage.

4. One-hot encoding. One-Hot Encoding is the process of creating dummy variables. Most of the categorical variable columns in the dataset are multi-class categorical variables encoded using integers. Many of the categorical features present in the data are not ordinal. This arrangement is not optimal for processing into a machine learning model as the model may mistakenly classify certain classes simply because their inherent integer number is higher. For example, the `disrict` column identifies a customer's location based on the district. There are 4 unique values for this column, [60, 62, 63, 69]. These numbers likely do not possess any additional meaning other than clustering clients based on their district. However, by passing this column into the models, the region of '63' will be considered to be higher than the region of '60' simply because 63 is greater than 60.

   To resolve this issue, we performed one hot encoding, creating binary columns for all unique values in a particular column. One-hot encoding essentially ensures that machine learning does not assume that higher numbers are more important. There were other fields in the dataset that were good candidates for one-hot encoding, such as `tarif-type`. However, we performed bucketisation for these fields instead of one hot encoding for other reasons, such as to reduce dimensionality, or to account for permutations and combinations of features that were not present in the training dataset. One-hot encoding was applied to the following columns in the dataset:

Column Name	**Description**
disrict	The district the client comes from.
client_catg	The category the client belongs to.
region	The region the client comes from.

5. Feature Normalisation. The data set contained mostly categorical features, with only one set of numerical features, that being the various consumption levels of the utilities in question (Gas or Electricity). Most of our dataset consists of binary encoded categorical variables, which ranges from 0 to 1. To prevent exploding gradients and vanishing gradients during our model training process later, we decided to normalise the numerical features using gaussian normalisation.

   We performed this for all 4 consummation levels. In addition, we took caution to ensure that the normalisation of datasets occurs after train test splitting to prevent data leakage. As we train our data, we are taking on an assumption that we are not aware of what is in our test data. It is with this intention that we perform feature normalisation or any feature engineering to our test data only.

FEATURE ENGINEERING

As part of our model creation process, our team performed a series of preliminary steps to engineer features first based on expert knowledge before conducting additional data exploration to find new

features. By basing our preliminary work on our hypothesis on the patterns and habits of fraudulent customers, we can speed up the Exploratory Data Analysis stage and generate more useful insights into our data.

The following is a description of features we engineered from the data, our methodology for obtaining these columns, as well as the justifications for including them as feature crosses in the final model:

1. District-Location Bucketising. We hypothesised that location could play a key part in identifying fraud customers. People coming from less developed neighbourhoods are more likely to commit crimes, as mentioned in a study conducted by Bucharest University of Economic Studies to detect electricity fraud in Tunisia (Oprea & Bâra, 2021). This could mean that the environment is a factor enabling or influencing others to commit cases of fraud or crime.

   Currently, the dataset contains two columns describing a user's location, `disrict`, the district a user comes from, and `region`, their region of origin. Within the training dataset, the `disrict` column contains 4 unique values, and `region` contains 25 unique values, for a total dimensionality of 29 after one hot encoding. In addition, after performing a group by function, we found 29 unique combinations of `disrict`, which is less than the total possible number of combinations of the two values, 36.

   The data does not capture 7 out of the 36 unique combinations of `disrict` and `region`, which means that any outliers that may appear from the test and validation set may severely affect the performance of the model. In addition, the dimensionality of 29 is fairly large. In order to reduce the dimensionality and avoid issues with missing regions and districts, we decide to categorise the districts and regions into two buckets each based on their fraud rate. Regions and districts with fraud rates higher than the sample fraud rate are classified 'high_risk_region' and 'high_risk_district' respectively, and all other regions and districts are classified low risk automatically. We decided to use mean instead of the median fraud rate as using the median fraud rate does not take into account the population differences in each region/district. New regions and districts with no data in the training set would automatically be classified into low risk. This means our model would be biased, but we believe it to be more reasonable for the model to be biased towards classifying new regions as low risk, since the lack of data on these areas could be evidence of low incidence of fraud.

2. Bucketising for Ordinal Features. The `counter_statue` feature represents the state of the counter, which could indicate that the counter was operational, faulty or otherwise. Based on our analysis, we found that clients operating utilities counters with an invoice history of multiple counter statuses had higher rates of fraud behaviour. Hypothetically, this would make sense: if a client is tampering with their utilities counter to commit fraud, then there would be a higher incidence of damage to the counter, leading to fluctuating counter statuses. To include this as part of our input matrix, we counted the number of different statuses a customer had for their counters throughout their invoice history. We created two additional binary features, `counter_status_low_risk` and `counter_status_high_risk`. Clients with only 1 unique counter status will have a `low_risk` value of 1. Clients with 3 or more different statuses would have a `high_risk` value of 1.

   We performed the same type of bucketising for `tarif_type` and `reading_remarque` since both displayed similar behaviours after running the same analysis.

In general, clients with multiple tariff types in their invoice history have higher risk of fraud. Hypothetically, this could be due to the client changing their geographic location or billing location to avoid arrest. Clients with multiple reading remarks in their history also had higher risks of fraud. Hypothetically, a fraud client who tampers with their counters will create variations in its functionality, which will lead to the engineer making more varied remarks in the client's invoice history.

3. Difference from Mean Consumption Trend. Based on data analysis, we found that fraud customers generally have median consumption values lower than the median of the dataset, and also had means which differed greatly from the mean of the dataset. This means the average fraudulent customer would likely have a significantly different gas and electricity consumption trend behaviour compared to the average/median behaviour. To take advantage of this, we performed a series of group by functions to find average consumption values by month, year and other features, such as district, region, tariff type etc. We then calculated the difference between the client's consumption level and the dataset's average for all invoiced consumptions, and passed the maximum, minimum and average difference into the input matrix as features.

   For this particular set of features, we only utilised `consummation_level_1` for the consumption data, since for the most part the other consumption columns from 2 to 4 were filled with 0 which meant we were unlikely to get any usable patterns for the model to learn.

We complete the feature engineering with a total of 28 features in our input matrix.

MACHINE LEARNING FOR ANOMALY DETECTION

The machine learning problem we are attempting to resolve in this situation is a binary classification problem. "Given a set of particular client data columns, identify if the client is a fraud or an honest customer." Due to the nature of the problem, we tried the following machine learning models to classify our data points:

1. Logistic Regression. For this particular method, we could pass the feature matrix into the `scikit-learn` LogisticRegression() object, using binary cross-entropy as the primary loss function. Logistic Regression would be a suitable model for this problem as it returns a binary variable, 0 or 1, which could be used to classify customers into fraud or non-fraud.

   The typical threshold is 0.5, but this can be changed depending on the severity of wrong classification. For example, the consequences of wrongly labelling a transaction as non-fraud is more severe than wrongly labelling one as fraud. We can lower the threshold, increasing the true positive rate at the cost of increasing the false positive rate.

   Logistic Regression is easy to implement and computationally efficient. It also has relatively few Hyperparameters, making it a stable, low variance model, and is easily explainable, which is good for business compliance. However, we expect Logistic Regression to rely heavily on the quality of feature engineering for model performance.

2. Decision Tree. We can also use `scikit-learn``'s DecisionTreeClassifier() to perform binary classification for our problem. This may be an effective means of classifying our input matrix as we have many binary features that could be used to segment the dataset. The performance of

the Decision Tree classifier can also be tuned by specifying maximum depth to improve its generalisability.

The Decision Tree Classifier generally runs quickly and is computationally efficient, but it is a high variance model that can vary wildly in performance. It is explainable, which could be useful in a business context, but can become quite complex for a high dimensional input matrix.

3. Artificial Neural Networks. We also tried using Keras to build an artificial neural network with a sigmoid activation layer as the output layer. Given the high dimensionality of the input matrix, we expected the decision boundary to be highly irregular, so ANN might be a good fit for the problem.

   ANNs can automatically engineer features, but for our dataset, complex data manipulation techniques needed to be applied to transform the consumption data, so we still expected our ANN to be highly reliant on our feature engineering.

MODEL METRICS

We looked at Accuracy, Precision and Recall in evaluating our models. We also plotted a count of True Positives, True Negatives, False Positives and False Negatives to verify our observations.

We determined that Accuracy alone may not be very useful due to the dataset imbalance and including both Precision and Recall would be useful in evaluating the ability of the model to isolate fraud samples often (former) and reliably (latter). Ideally, our model would have good accuracy, precision, and recall, but we would prioritise recall, followed by precision and then accuracy, since in our case of fraud detection, we are more concerned with false negatives because it means we fail to identify fraudulent cases, which could lead to losses for the company.

MODEL ARCHITECTURE AND HYPERPARAMETER TUNING

Logistic Regression

For logistic regression, the hyperparameter we tuned was the cut-off threshold for confidence to classify a client as a fraud. By default, the threshold is 0.5. After fitting a logistic regression model, we performed a grid search of 41 equally spaced threshold values between 0 and 1 and plotted a graph of varying metrics to see the impact of altering the threshold. Our observations as follows:

1. Accuracy and Recall have an inverse relationship. As threshold increases from 0 to 1, accuracy increased, and recall decreased. At low threshold, most samples with low confidence would be labelled as fraud, which would lead to higher recall as more actual fraud customers were correctly labelled, however most honest customers would also be labelled as frauds.

2. Accuracy and recall flatten at around threshold 0.8, with accuracy at a value of 0.94, the natural rate of fraud, which suggests that past that threshold most samples would have been labelled as honest. This is supported by the number of True Negative and False Negative counts peaking at 0.8.

3. F1 Score peaks between 0.6 to 0.7, where Accuracy and Recall intersect at around 0.6 each. This suggests that the model has the best balance of precision, recall and accuracy at a

confidence threshold of 0.6. As such, our final regression model will use a confidence threshold of 0.6.

Decision Tree

For the Decision Tree, we performed grid search varying max depth from 1 to 30, and changing the error function (entropy vs gini). We chose the absolute max depth of 30 as we wanted the number of splits to be less than or near to our input dimensionality of 28. Using grid search, we found that Entropy generally performed better on accuracy, recall and precision, as such we stuck with Entropy as our error function. Generally, test accuracy improved with tree depth, but all other metrics worsened with tree depth. Interestingly, the best performing model is a shallow model of depth 2, with the highest F1 score and higher than average Accuracy score. As such we decided to stick with a shallow decision tree of 2.

Artificial Neural Network

For our model architecture, due to the overwhelming number of hyperparameters to train, we used randomised grid search to optimise our model. For the activation functions, we decided to go with sigmoid for the output layer to suit the nature of the problem, binary classification. For input and hidden layers, we use ReLU as we had multiple numerical columns with domains exceeding 0 and 1.

We randomly initiated 100 models with layers no more than 10 and neurons per layer no more than 50. We plotted scatterplots to determine the relationship of model complexity on various metrics. Except for a few outliers, our models generally performed similarly, with F1 Scores around 0.15, accuracy around 0.5, recall around 0.8 and precision around 0.08, regardless of model complexity.

Due to the difficulty in visualising metrics by hyperparameters, we ordered our output by F1 Score and selected our model greedily. Out of the top 5 models presented, we selected the least complex model, defined as the model with the least number of layers, followed by the least number of neurons. Due to random initialisation, our hyperparameters differed across iterations. However, our investigation reveals that most of the models performed similarly regardless of depth and neuron count, as such we were not too concerned with the nature of the final hyperparameters used.

VALIDATION SET RESULTS AND FEATURE IMPORTANCE

After tuning our hyperparameters, we merged the training and test set data, rebalanced the merged dataset, and trained our selected models with the merged data before final evaluation with our validation set. All values to 3 d.p. The results are as follows:

Model	Hyperparameters	Acc	Recall	Precision	F1
ANN	1 Hidden Layer, 32 Neurons, ReLU	0.445	0.874	0.081	0.148
DT	Max Depth 2, Entropy Loss	0.675	0.549	0.092	0.157
Logistic	Threshold = 0.62	0.707	0.561	0.103	0.175

We also used the `shap` module to display the feature importance of our dataset on our logistic regression model. The full results can be found in the notebook accompanying this report.

BUSINESS USE CASES

Our model is a binary classification model that has been used to classify samples into fraud and non-fraud based on numerical and categorical characteristics. Apart from fraud detection in electricity consumption, there are numerous other applications of binary classification models, such as detecting frauds in banking, document forgery, and email spam. In the case of our project, this fraud detection methodology could be used to pre-filter new customers for the business to allocate human staff or Police for additional supervision.

CONCLUDING STATEMENT AND IMPROVEMENTS TO METHODOLOGY

In hindsight, our model performances were less than ideal. Throughout our model tuning and feature engineering process, we were unable to generate a model that could produce a result that has high accuracy, precision and recall all at an acceptable level. Often, we found ourselves trading one metric (for example, recall) at the expense of the other two. We considered a few reasons for the underperformance:

1. Hard to verify Dataset Cleanliness. The dataset we sourced was generally complete, but not necessarily clean. We resolved the issues with `reading_remarque` in the invoice dataset, but we were unable to verify if the dataset was completely clean as all columns were encoded with integers with no contextual explanation provided by the utilities company (we suspect this is due to customer privacy reasons).

2. Heavily imbalanced Data. We already knew that the dataset had more honest than fraud customers. But we found out later that the data was also imbalanced on a categorical level. It was difficult to utilise the numerical data provided due to categorical imbalance.

   The raw consumption data could be grouped by many different categories (client_catg, region, district etc.), month and year. As a result of this excessive grouping, some groups have many data points and some only have a few, which could lead to problems with representation. For example, a particular customer may have many invoice data points for all months to map a trend, but another customer may have only 1 or 2 data points in their invoice history, which may not be sufficient to accurately describe the client's behaviour and give an accurate classification.

The dataset was difficult to work with and required better data and extensive feature engineering to improve its output. Due to limitations in resources and time, we were unable to do both.

To conclude, our team utilised a series of pre-processing and feature engineering steps to develop an input dataset, which we passed into Logistic Regression, Decision Tree and Artificial Neural Network models in order to detect anomaly behaviour in utilities consumption.

APPENDIX

Chew, H. M. (2016, July 8). Another phone scam now targets SP Services customers; police report made. The Straits Times. https://www.straitstimes.com/singapore/another-phone-scam-now-targets-sp-services-customers-police-report-made

GOV.SG. (2021, June 20). False and misleading websites that claim that Singaporeans are overpaying for electricity. http://www.gov.sg/article/false-and-misleading-websites-that-claim-that-singaporeans-are-overpaying-for-electricity

Oprea, S.-V., & Bâra, A. (2022, February 28). Feature engineering solution with structured query language analytic functions in detecting electricity frauds using Machine Learning. Nature News. Retrieved October 15, 2022, from https://www.nature.com/articles/s41598-022-07337-7

Oprea, S.-V., & Bâra, A. (2021, July 28). Machine learning classification algorithms and anomaly detection in conventional meters and Tunisian electricity consumption large datasets. Computers & Electrical Engineering. Retrieved October 15, 2022, from https://www.sciencedirect.com/science/article/abs/pii/S0045790621003013#preview-section-references

Zheng, Z., Yang, Y., Niu, X., Dai, H.-N., & Zhou, Y. (2018). Wide and deep convolutional neural networks for electricity-theft detection to secure smart grids. IEEE Transactions on Industrial Informatics, 14(4), 1606–1615. https://doi.org/10.1109/TII.2017.2785963

Ramos, C.C., Rodrigues, D., de Souza, A.N., & Papa, J.P. (2018). On the Study of Commercial Losses in Brazil: A Binary Black Hole Algorithm for Theft Characterization. IEEE Transactions on Smart Grid, 9, 676-683.

Jain, S., Choksi, K. A., & Pindoriya, N. M. (2019). Rule-based classification of energy theft and anomalies in consumers load demand profile. IET Smart Grid, 2(4), 612–624. https://doi.org/10.1049/iet-stg.2019.0081

Oprea, S.-V., Bâra, A., Puican, F. C., & Radu, I. C. (2021). Anomaly detection with machine learning algorithms and big data in electricity consumption. Sustainability, 13(19), 10963. https://doi.org/10.3390/su131910963

Buzau, M. M., Tejedor-Aguilera, J., Cruz-Romero, P., & Gómez-Expósito, A. (2019). Detection of non-technical losses using smart meter data and supervised learning. IEEE Transactions on Smart Grid, 10(3), 2661–2670. https://doi.org/10.1109/TSG.2018.2807925

Amara Korba, A., Tamani, N., Ghamri-Doudane, Y., & karabadji, N. E. I. (2020). Anomaly-based framework for detecting power overloading cyberattacks in smart grid AMI. Computers & Security, 96, 101896. https://doi.org/10.1016/j.cose.2020.101896

Nagi, J., Yap, K.S., Tiong, S.K., Ahmed, S.K., & Mohamad, M. (2010). Nontechnical Loss Detection for Metered Customers in Power Utility Using Support Vector Machines. IEEE Transactions on Power Delivery, 25, 1162-1171.

Massaferro, P., Martino, J. M. D., & Fernandez, A. (2020). Fraud detection in electric power distribution: An approach that maximizes the economic return. IEEE Transactions on Power Systems, 35, 703–710. https://doi.org/10.1109/TPWRS.2019.2928276