SGCC Electricity theft detection using deep learning model

**Introduction**

Electricity theft is a critical issue that affects the sustainability of power distribution systems. Electricity theft can be categorized into two types: technical and non-technical losses. Technical losses are due to the inherent inefficiencies in power transmission and distribution systems, whereas non-technical losses are caused by consumers' intentional tampering with the power distribution infrastructure.

Utilities have traditionally relied on manual inspections, meter readings, and advanced metering infrastructure to combat electricity theft (AMI). These methods, however, have limitations in terms of accuracy, time, and cost, prompting the development of alternative solutions. Deep learning models, which can effectively detect fraudulent behavior in power consumption patterns, are one promising approach.

The study follows a rigorous methodology that involves data pre-processing, feature extraction, and model training. Then I evaluated the performance of the LSTM and CNN-LSTM model on the SGCC dataset,using various performance metrics such as precision, recall, and F1 score**.**

The rest of this report is organized as follows. Section 2 describes the SGCC dataset and its characteristics. Section 3 details the methodology used for data pre-processing, feature extraction, and model training. Section 4 presents our results and evaluation of the model's performance.

**SGCC Dataset**

This section describes the dataset's attributes, which were obtained from the State Grid Corporation of China (http://www.sgcc.com.cn/). Data was collected for 34 months, that is from 2014-01-01 to 2016-10-31. For the investigation, SGCC provides real-time energy usage data and is made up of 42,372 rows and 1,035 columns. The number (42372) refers to the statistics of total energy consumers' usage for a period of 1,035 days. The customer ID is in the first column, a prediction indication called "Tag" is in the second column, and the day's columns begin in the third column (1035). The information in the flag column (zero and one) identifies the various customer types (normal or fraudulent). The average amount of energy consumed is represented by the total number of zeros in the "Flag" column, which is 1. (38757). While "the flag" only mentions one thief, there are several (3615). Data on electricity usage is collected by smart meters and the data network collects information. The smart meter, sensor, or data transmission server may fail to store, resulting in missing or incorrect data in the electricity consumption statistics. If the missing duplicates are removed, the data set shrinks, making analysis more difficult. Thus we discuss in the next section how to effectively deal with missing values in the dataset.

**Overview of the electricity consumption**

I conducted a preliminary analysis of electricity consumption data through the distribution of electricity consumption over a month in the dataset for both normal and fraudulent consumers. It can be seen from Figure (3) that there is a fluctuation in the electricity consumption data day by day. It is hard to capture the key characteristics of electricity thieves and normal customers from this 1-D data.

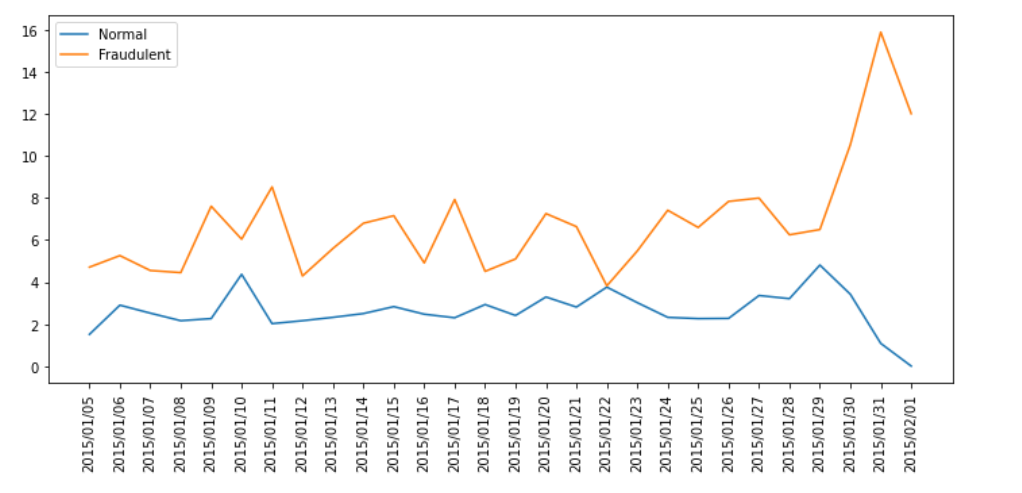


Figure (1)

Figure (1) shows the distribution of electricity consumption by an honest customer over 4 weeks. As can be seen, there is some pattern as for 3 weeks the highest recorded electricity consumption happens on Thursday and the range for all three weeks is more or less the same. This pattern may be evidence of the honesty of the consumer as humans have fixed weekly schedules which would make sense for them to have a pattern of electricity consumption over 4 weeks.

Figure (2) shows the distribution of electricity consumption by a fraudulent customer over 4 weeks. As can be seen, there is no particular pattern in the data and it follows a random distribution. This randomness may be proof of tampering with the meter readings.

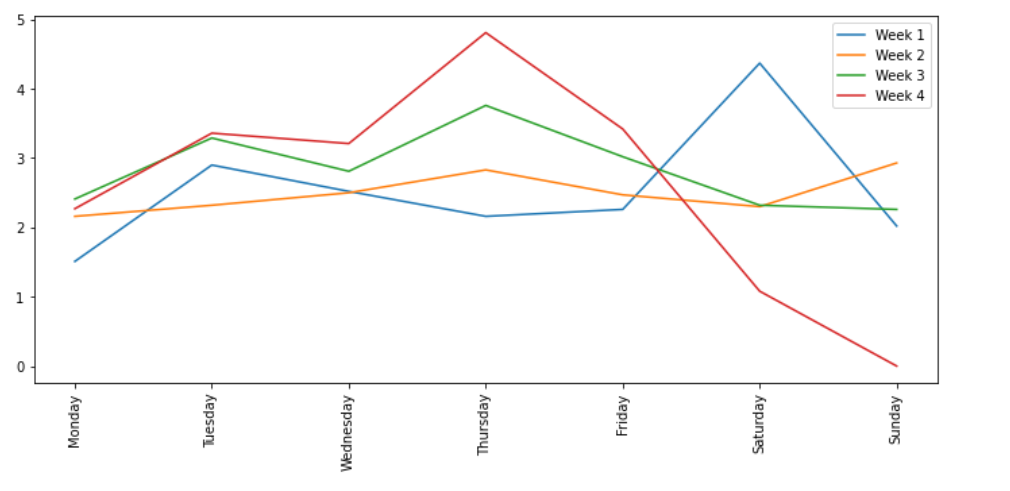


Figure (2)

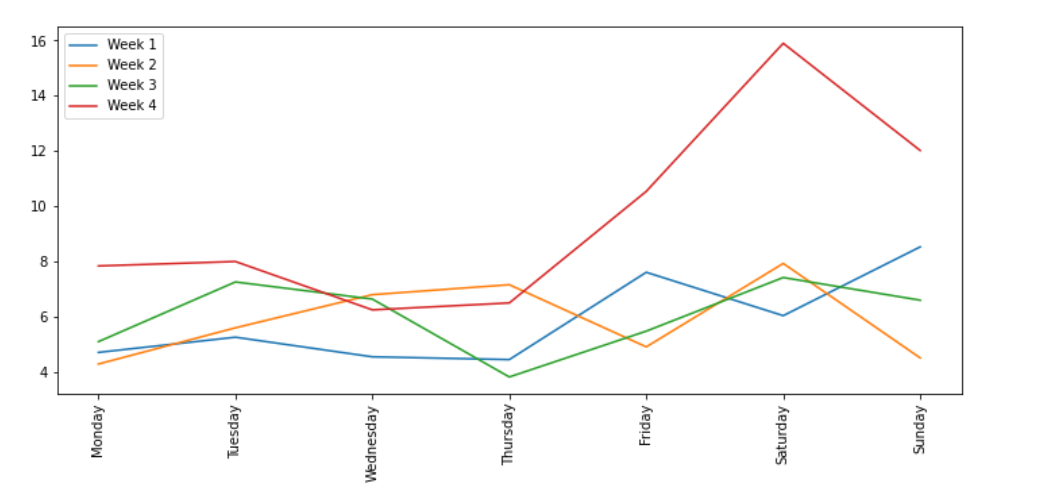
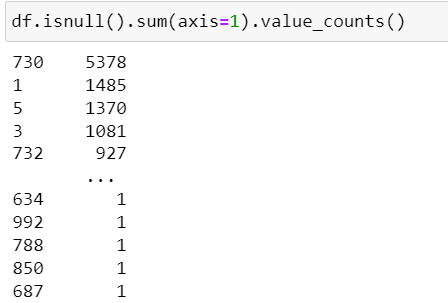


Figure (3)

**Data Pre-Processing**

1. Handling missing values

The 42372-row-long dataset had missing values as high as over 730 missing values in one row with 1036 values. In the below code snippet’s output, the first column shows the number of missing values and the second column shows the number of rows with that number of missing values.



To solve this issue of missing values, the value for these cells was imputed with the below-proposed preprocessing algorithm which utilized the local average value of the consumed power to calculate the missing values. If there was a missing value or NaN value at the position x\_i, the value was computed as

f(xi) = LocalAverage if xi = NaN

xi if xi ≠ NaN.

The local average was given as

LocalAverage = f(xi)

Some more conditions were also added, that is if

f(xi) = NaN

then xi would be 0. Further for values where there are no preceding 5 values to calculate the local average of, the mean of available values was taken and inserted in the place of missing values. Doing so, there were no missing values left in the dataset.

1. Handling Outliers and Normalizing the Dataset

The electricity consumption data was also found to have erroneous values (i.e., outliers) in the. In particular, we restore the value by the following equation according to the “Three-sigma rule of thumb”.

f(xi) = avg(x) + 2 · std(x) if xi > avg(x) + 2 · std(x),

xi otherwise,

This method can effectively mitigate the outliers.

After dealing with the missing values and the outliers, we need to normalize the electricity consumption data because the neural network is sensitive to diverse data. In particular, we choose the MAX–MIN scaling method to normalize the data according to the following equation:

f(xi) =

1. Balancing the Dataset

Any real-world data set used for classification is almost certainly imbalanced, with the event of interest being extremely rare (minority examples) and non-interesting events dominating the data set (majority examples). Similarly, in our case, there are 38757 normal customers but only 3615 fraudulent customers. As a result, the machine learning models we create will be biased. Consider this: if you feed the model only 0 for every possible combination, it will return a 0 for every set of inputs. Thus we use ADASYN to avoid this problem. ADASYN (Adaptive Synthetic) is a synthetic data generation algorithm whose main advantages are not copying the same minority data and generating more data for "harder to learn" examples. ADASYN is a more general framework; for each minority observation, it first determines the impurity of the neighborhood by taking the ratio of majority observations in the neighborhood and k. This impurity ratio is converted into a probability distribution by making the sum 1. The higher the ratio, the more synthetic points are generated for that specific point.

Using the old classifier, 13 out of 2064 would be classified as a minority. Minority class samples are generated using ADASYN such that there are now 38757 regular customers but only 5678 fraudulent customers.

**CNN-LSTM model**

For this dataset I have used a CNN-LSTM model, depicted in Figure (4) in which the CNN layers precede the LSTM layers. It is highly efficient and robust when used for smart grid data classification.

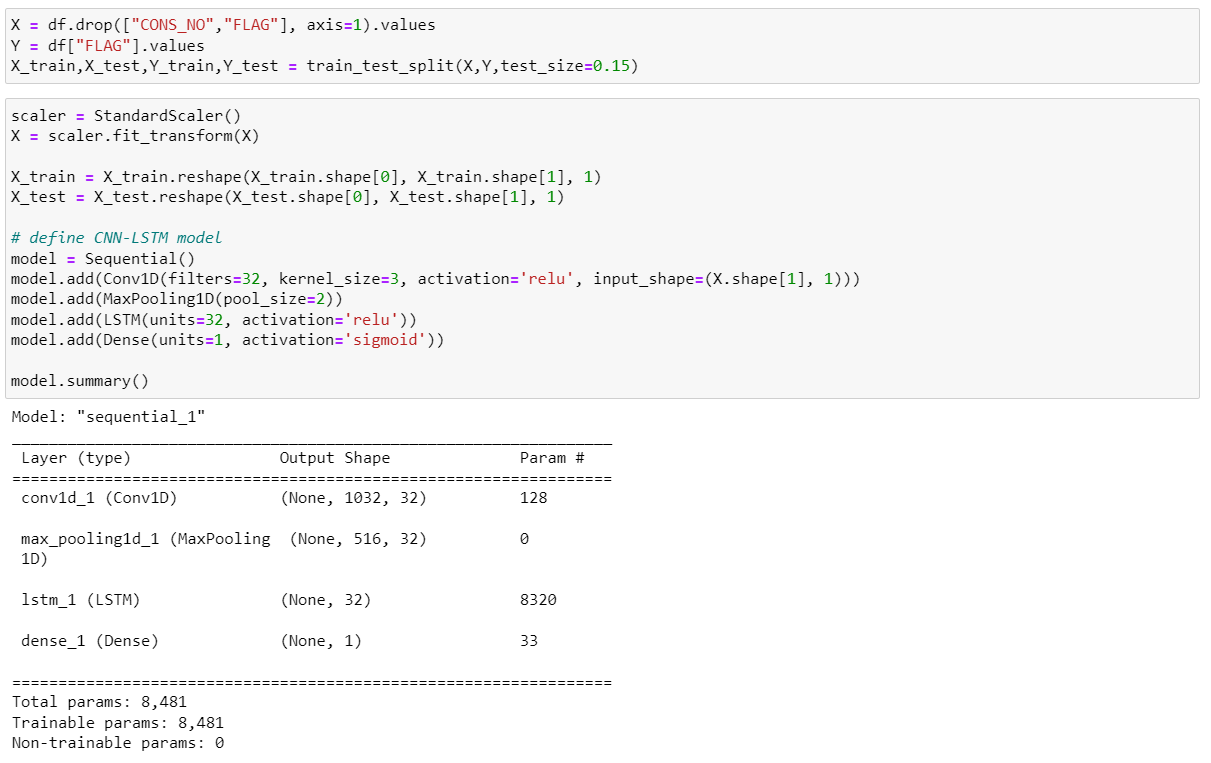


Figure (4)

The CNN-LSTM model is made up of two main parts: a CNN and an LSTM. CNN is used to extract features from the input time series data, whereas LSTM is used to capture temporal dependencies and predict them.

The model's input is a data frame containing electricity consumption data for 32746 customers over 1034 days, which has been pre-processed by computing missing values and removing outliers. The dataset is then divided into features and labels, which are further divided into train and test data by an 85%-15% ratio. The X train and X test data are then standardized using the scikit learn library's StandardScaler. The data is then reshaped to have a third dimension of one, which is required for the CNN.

The model's CNN component is a 1D convolutional layer with 32 filters and a kernel size of 3. The convolutional layer output is fed into a max pooling layer with a pool size of 2. The convolutional layer extracts local features from the input time series data, whereas the pooling layer reduces the dimensionality of the feature maps and adds translation invariance to the model.

The CNN's output is then fed into a 32-unit LSTM layer. The LSTM layer is used to capture and predict temporal dependencies in data.

The output of the CNN is then fed into an LSTM layer with 32 units. The LSTM layer is used to capture and predict temporal dependencies in data. The LSTM layer's output is routed to a dense output layer with a single unit and a sigmoid activation function. The sigmoid activation function is used to compress the output between 0 and 1, which represents the likelihood of electricity theft.

The model is trained with the loss function "binary crossentropy" and the optimizer "Adam." The accuracy of the trained model is reported after it has been evaluated on the test data.

**References**

This project has been done taking inspiration from the below papers:

1. <https://medium.com/@ruinian/an-introduction-to-adasyn-with-code-1383a5ece7aa>
2. <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9225572>
3. <https://www.mdpi.com/1996-1073/12/17/3310>