AI Project Report

on

Energy Theft Detection using Deep Learning

Under the guidance of

Dr. Abdul Wahid

Student Name: -

ADITYA SUSHIL TIWARI (20BCS007)

Contents

INTRODUCTION

Energy theft also known as electricity or power theft is a significant worldwide issue. Not only does it pose financial losses to utility providers but also disrupts fair distribution and management of resources. To address these challenges, the development of robust fraud detection systems leveraging advanced machine learning techniques has become imperative.

This project aims to design and implement a Deep Learning-Based Fraud Detection System for Electricity Consumption. The system utilizes Neural Networks to analyze consumption patterns and identify potential instances of fraudulent behavior. Leveraging historical consumption data, the system learns complex patterns and anomalies indicative of fraudulent activities, enabling utility providers to take proactive measures to mitigate risks and maintain the integrity of their services.

We begin by cleaning and preprocessing the dataset, which includes dropping duplicates, handling null values, treating outliers and more, to prepare it for model training. Subsequently, we experiment with different neural network models, to identify the most effective approach for fraud detection.

Through extensive experimentation and evaluation, we assess the performance of each model in terms of Accuracy, F1 score for each classification, and Area under the ROC Curve (AUC) when trained for different numbers of epochs. We analyze the impact of different numbers of layers and neurons in the neural networks, evaluation metrics and thresholds on the model's efficacy.

Overall, this project contributes to the advancement of fraud detection techniques in the energy sector, offering insights and methodologies that can be leveraged by utility providers and researchers to combat fraudulent activities effectively.

Data Cleaning and Preprocessing

Effective data cleaning and preprocessing steps are crucial to ensuring the quality and accuracy of the machine learning models. In this project we load the raw data into a dataframe from a CSV file using pandas. The features and labels are separated into different dataframes for more convenient handling of data. Duplicate rows are dropped from both the label and feature dataframes. The column names in the feature dataframe are converted to datetime format and sorted in chronological order.

For handling missing values (NaNs), linear interpolation is used, with a limit of filling up to 2 consecutive missing values in both forward and backward directions along each row in the feature dataframe. After interpolation any remaining missing values are replaced with 0. Linear interpolation provides an estimate of the missing values by considering the trend and continuity in data and attempts to minimize the impact of interpolation on the datasets statistical properties. Filling with 0 after interpolation ensures the dataset doesn't contain any missing values which can be problematic for many machine learning algorithms and data processing steps. Filling with 0 is a straightforward way to handle the residual missing values, especially if having zeroes does not significantly distort the analysis or the modeling results.

Following the handling of null values, rows with all zero values are removed from the feature dataframe and the corresponding rows in the label dataframe are removed as well. These rows may indicate excessive missing or invalid values, distort statistical measures of the data and lead to poor model performance.

For handling outliers, the outliers in each row of the feature dataframe are replaced with **(m + 3 * sd)**, where **m** is the mean of values in the row and **sd** is the standard deviation of values in the row. This is done to reduce the impact of extreme values on statistical measures and machine learning models. Removing outliers can lead to loss of information, therefore they are capped to ensure that the data remains largely intact while mitigating the adverse effects of extreme values.

The feature data is then transposed and normalized by applying the Min-Max scaler along the columns which are the rows (instances or data points) of the original data and then transposed again to its original form. This operation scales all values in a row (which corresponds to a consumer's data over time) from 0 to 1. Our scaled data is now converted into a CSV file named **preprocessedR.csv** for ease of sharing.

Preparing Training and Test Datasets

For preparing the training and test datasets for training and evaluating our neural networks we use the **read_data()** function. The function reads the preprocessed data from its CSV file and extracts the feature (**X**) and label (**y**) data. Using the label data, some statistics like the total number of normal and fraud consumers are printed out. The column names of the feature data are converted to datetime objects and sorted chronologically.

We then pass the feature and label data to scikit-learn's train-test split function with test size 20% and random state equal to 0. The function returns the train and test datasets for both features and labels. The percentage of normal consumers in the test set is printed out.

As we set the **oversampling_flag** equal to 1, Synthetic Minority Over-Sampling Technique (SMOTE) is applied on the training dataset to address its imbalance, as it has very few data points of fraud customers. The **sampling_strategy** is set to 0.2, so as to increase the percentage of fraud customers in the training dataset to 20%. This is done by generating synthetic samples for the minority class. After the oversampling is performed on the training dataset, some statistics of the same are printed out.

The function ends by returning the training and test datasets of both the feature and label data: **X_train, X_test, y_train, y_test** .

Evaluation Function for Neural Networks

This function, named **results**, evaluates the neural network's predictions using the test set of the label data, using several metrics like Accuracy, AUC (Area Under the ROC Curve) and F1 scores for classification of each class. The values of these metrics are printed out. The function also prints out a confusion matrix based on the neural network's predictions and the actual values in the test set.

The function ends by returning the values of the various metrics calculated by it: - **auc, accuracy, f1_score_0, f1_score_1** .

Neural Network 1

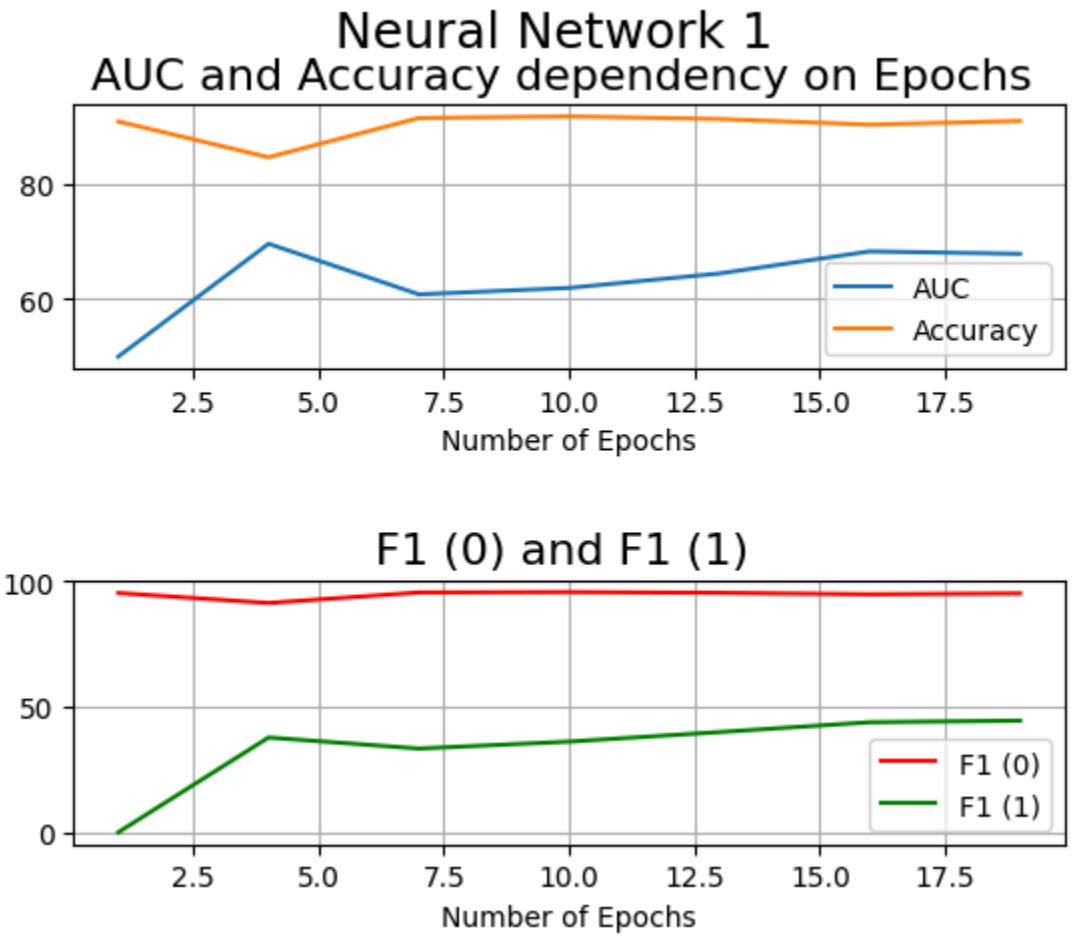
We define a function **NN1** for defining, training and evaluating our first neural network multiple times over different numbers of epochs. This function takes the training and test sets of the training and test data returned by the **read_data()** function.

We use a sequential neural network, with an input layer having 1034 neurons, 3 hidden layers with 100 neurons each, 1 hidden layer with 10 neurons and an output layer with one neuron. All neurons use the **ReLU** activation function except the output neuron which uses the **Sigmoid** activation function, suitable for binary classification. All the layers are dense (fully connected).

The model is then compiled, while specifying **binary cross-entropy** as the loss function, **adam** as the optimizer and **accuracy** as the evaluation metric.

After compiling the model we train the model on the training dataset for different numbers of epochs ranging from 1 to 19. While generating predictions the output of the model is converted to binary predictions (0 or 1) using a threshold of **0.5** . The predictions are passed to the **results** function for evaluation which returns the model's performance in each training round and prints out a confusion matrix for the same. The returned performance metrics are recorded in a CSV file for visualization purposes.

On visualizing the performance of the model after being trained for different number of epochs, we get the below graph: -



We can infer that the accuracy and the F1 score of classification of zero are high throughout the different training rounds (a drop noticed at round with 4 epochs). The values for AUC and F1 score of classification of 1 rise till the round with 4 epochs but have a drop thereafter and gradually stabilize by the round with 19 epochs.

Neural Network 2

We define a function **NN2** for defining, training and evaluating our second neural network multiple times over different numbers of epochs. This function takes the training and test sets of the training and test data returned by the **read_data()** function.

We use a sequential neural network, with an input layer having 1034 neurons, 1 hidden layer with 500 neurons, 1 hidden layer with 250 neurons, 1 hidden layer with 125 neurons, 1 hidden layer with 10 neurons and an output layer with one neuron. All neurons use the **ReLU** activation function except the output neuron which uses the **Sigmoid** activation function, suitable for binary classification. All the layers are dense (fully connected).

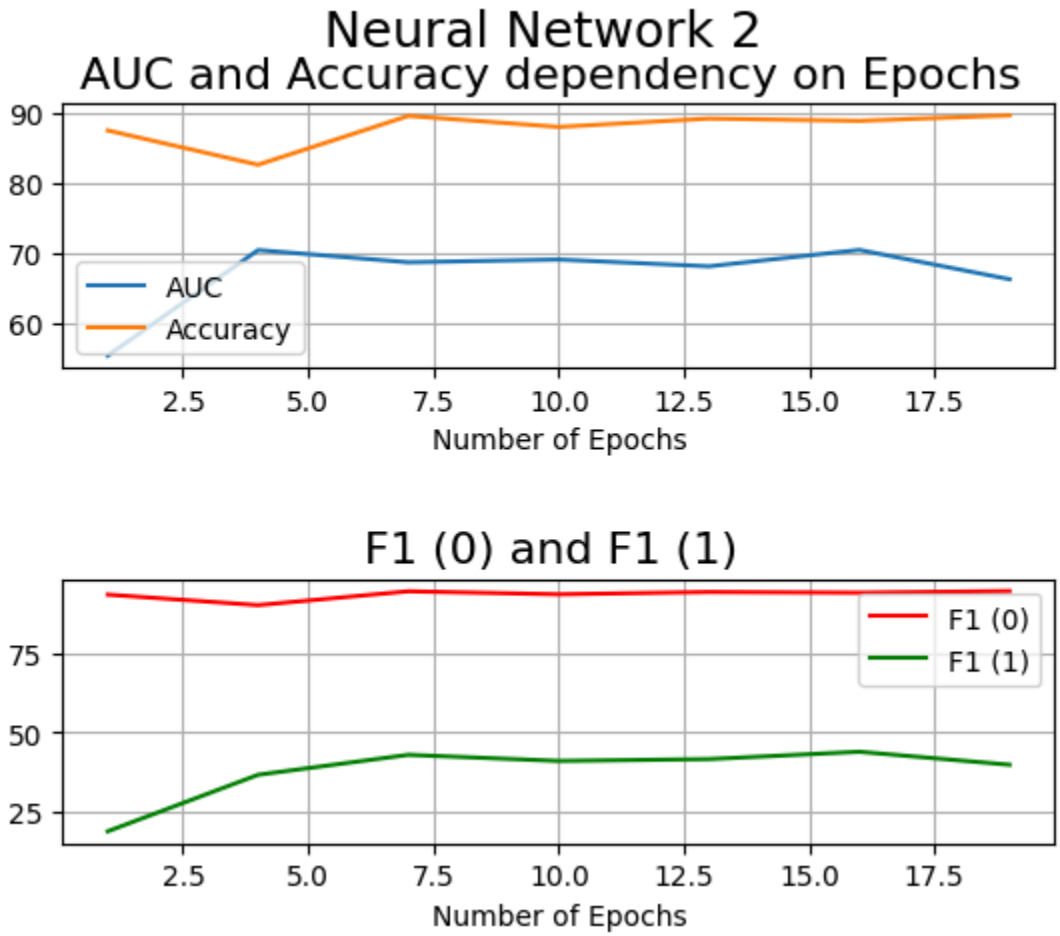
The model is then compiled, while specifying **binary cross-entropy** as the loss function, **adam** as the optimizer and **binary accuracy** as the evaluation metric.

After compiling the model we train the model on the training dataset for different numbers of epochs ranging from 1 to 19. While generating predictions the output of the model is converted to binary predictions (0 or 1) using a threshold of **0.3** . The predictions are passed to the **results** function for evaluation which returns the model's performance in each training round and prints out a confusion matrix for the same.

The returned performance metrics are recorded in a CSV file for visualization purposes.

(continued on following page)

On visualizing the performance of the model after being trained for different number of epochs, we get the below graph: -



We can infer that the accuracy and the F1 score of classification of zero are high throughout the different training rounds for this neural network as well (a drop noticed at round with 4 epochs). The values for F1 score of classification of 1 rise till the round with 7 epochs and are stable thereafter. The values for AUC rise till the round with 4 epochs and are mostly stable thereafter.

Conclusion

We can conclude that both neural networks have a high overall accuracy and are good at classifying normal consumers but are poor at classifying fraudulent consumers. For both models the AUC score and the F1 score for classification of fraudulent consumers increases with the number of epochs but stabilizes by the round with 19 epochs. The second model, compared to the first, has more neurons, a binary accuracy metric and a lower threshold of 0.3 . It shows a significant improvement in the AUC score but the remaining metrics show mostly the same trends as those for the first model.