A cognitive security framework for detecting intrusions in IoT and 5G utilizing deep learning

This contains Python scripts that we've developed to preprocessing data, particularly focusing on scaling, label encoding, and handling class imbalance using techniques like SMOTE and RandomUnderSampler. The code is designed for use in machine learning , particularly classification tasks.

Code Description

1. Importing necessary libraries: sklearn.preprocessing.OneHotEncoder, LabelEncoder, and MinMaxScaler for data preprocessing. imblearn for handling class imbalance.Counter for counting occurrences of each class.

2. Data Loading and Exploration With the help of Pandas, it reads CSV files which includes CICIDS-2017, 2018 and UNSW-NB15 datasets, aggregates data frames, and provides an initial overview of the dataset distributions.

3. Visualizing: After the importing the data visualizinr the data that has the classes types of the attack.

```
 BENIGN                        2271320
DoS Hulk                       230124
PortScan                       158804
DDoS                           128025
DoS GoldenEye                   10293
FTP-Patator                      7935
SSH-Patator                      5897
DoS slowloris                    5796
DoS Slowhttptest                 5499
Bot                              1956
Web Attack � Brute Force         1507
Web Attack � XSS                  652
Infiltration                       36
Web Attack � Sql Injection         21
Heartbleed                         11
```

4. Preprocessing Data: Converting all classes into two classes whether attack or not.Scaling the features using MinMaxScaler. Encoding categorical labels using LabelEncoder.

5 . Class Imbalance Handling: Utilizing SMOTE (Synthetic Minority Over-sampling Technique) for oversampling minority classes.Using RandomUnderSampler for undersampling majority classes.

6 . After the Preprocessing Visualizing the dataset After class balancing, the instances per label in test set are as follows-

```
 BENIGN          567830
attack          139139
```

Printing the shape of the training and testing sets after preprocessing.Displaying the distribution of instances per label in both the training and testing sets.Creating a dictionary mapping class labels to numerical values.

7. RGB and grayscale images:

a). This demonstrates how to convert numerical features into visual representations for analysis and visualization purposes.

b). Normalizes input features using min-max normalization to scale values between 0 and 1.

c). Converts normalized features to RGB images by mapping them to 24-bit numbers and extracting RGB components.

d).Converts normalized features to grayscale images by scaling values up to 255.

e). Combines normalization and mapping functions to generate RGB and grayscale images from input data.

8.Pre-trained Models Conversion to TensorFlow.js: This code provides a simple method to convert pre-trained models saved in the HDF5 format (.h5) using Keras into the TensorFlow.js format.

```
 There are two models:
model_gray (1).h5
model_rgb (1).h5
```

9. Implementation of GUI: After saving the models we implemented the GUI for the Models. That consist two parts:

1. Frontend
2. Backend

```
 1.Frontend: For the Frontend we use HTML,CSS.
2.Backend: Used the flask and python.
```

10. Source Directory Tree:

```
IT352_Project/
├── main.py
├── model_train.ipynb
├── output/
│   └── prediction.txt
├── models/
│   ├── model_gray (1).h5
│   └── model_rgb (1).h5
│   └── model_rgb.h5
│   └── model_rgb.h5
├── scaler.joblib
├── static/
│   └── images/
│       ├── generated_gray.png
│       └── generated_rgb.png
└── templates/
    └── index.html
```

Authors

```
 Shuaib jawid - 211IT087
Ashwani kumar - 211IT013
Jaheer khan - 211IT026
```