**Challenge** **Question 2: Linux malware detection and classification (LMDC)**

**Data Pre-Processing**

• The dataset contains benign ELF (Executable and Linkable Format) files and five categories of malware ELF files: Backdoor, Botnet, DDoS, Trojan, Virus

• We parse through each of the files provided in the dataset using python elftools functions. In doing this we first observe the file magic number, then figure out whether the corresponding file is an ELF file or not and only keep ELF files for the dataset

• We extract the binaries from “.text” section which contain the program code of the file to capture the opcode patterns inherent to malware and benign files that may help the classification task between benign and malware

• We observe “.text” section may not present in some ELF files. To capture meaningful features for such files we also extract binaries from ".comment", ".note", ".symtab", ".strtab", ".rodata", and “.init” sections which may contain anomalous patterns inherent to each malware class that may help to distinguish between malware types

**Feature Extraction**

• 80% the dataset is used as the training set and 20% of the dataset is used as the testing test

• Given a file, we extract top 75 character-level (1,2 ,3)-gram TF-IDF features corresponding to each file section binaries collected during the data pre-processing step to capture the nibble/byte patterns inherent to files

• A file is represented by the concatenatd feature vectors corresponding to each of its file section

• To extract character-level (1,2 ,3)-gram TF-IDF features, we use TF-IDF vectorizer and its fit\_transform functionality on the training dataset

• To generate the same features for test set we use the trained TF-IDF vectorizer and its transform function

**Training**

• We train several supervised learning classification algorithms that include SVM, Logistic regression, Random Forest, and Multi-layer perceptron using the train set prepared above

**Testing**

• We find XGboost provide better F1 scores across all the classes and a better accuracy

• We additionally tried several ensemble methods that includes bagging, boosting and stacking. These ensemble methods did not increase the accuracy and F1 scores by more than 2%. Therefore, we decided to choose XGboost as our ML model (computationally more efficient compared to other ensemble methods and provide good performance)

**Techniques used for evaluating the quality of features and models**

• We visualized the feature representation of our training test using 2-d tsne method to see how much separation we achieve between each class

• We used oversampling techniques such as SMOTE and SOUP to generate synthetic data for the minority classes (Malware classes) and train the classification models using the resulting balanced dataset and observe the F1 scores and accuracy

**Folder Guide**

folder |

-- LMDC\_test.py |

-- File1\_prob2.ipynb |

-- result.csv |

-- XGB\_model |

-- trans-.text.pkl |

-- trans-.comment.pkl |

-- trans-.symtab.pkl |

-- trans-.strtab.pkl |

-- trans-.rodata.pkl |

-- trans-.init.pkl |

**Dependencies**

• elftools; os; sklearn; sys; numpy; collections; random; pickle; multi\_imbalance; imblearn