Name: Gautam Vira
Student ID: 300302217

Assignment 3 – CSI – 5388

Data analysis and Preprocessing

The Static_dataset.csv is imported and read as a dataframe, Any infinity or -infinity values are replace with null values. The dataframe is then checked for any null values; the column 'longest_word' contains some null values. The rows containing null values are dropped as the rows containing null values are insignificant compared to the size of the dataframe.

After the dataset is clean, the target variable is analyzed to see the number of points belonging to each class. 147179 points belong to the class 1 and 120887 points belong to the class 0 – this indicates that there is not a class imbalance (see fig. 1). The dataset contains a feature called 'timestamp', an attempt is made to sort the dataset based on this feature to simulate a real environment in which case the starting (older) rows would be used to train the models and the later (newer) rows would be used to validate the models, however, it was found that the timestamp did not contain significant details which could help in sorting the dataset in an accurate manner as the timestamp only contained MM:SS which did not have enough unique values, and so this feature was dropped.

The feature 'longest_word' contains strings of different lengths so the feature values were replaced by the length of the values (strings) which could be used to train the models. Also, the feature 'sld' contained string values which were hashed to convert them from strings as well as encode them to be used in the models.

The dataset is split into X and y and an initial Mutual Information score is used to analyze the dependence of the input variables to the target variables (see fig. 2) – features like 'sld', 'labels', 'special', 'entropy', 'FQDN_count' have a high score for their dependence. Additionally, a correlation heatmap is also plotted to further investigate the correlation of the input features with the target variable which also gave similar results (see fig. 3). No features are dropped based on these observations yet.

Model Selection, Hyperparameter Tuning, and Evaluation

A train-test split is performed initially so that the models selection and hyperparameter tuning can be performed on the train set leaving the test set for evaluation. The two models chosen for evaluation are Random Forest Classifier and Gradient Boosting Classifier as ensemble and boosting methods perform really well for binary classification problems.

Two pipelines and two randomized searches are used for each model: each pipeline contains a dictionary of parameters to be tested for each model and a feature selection technique. Each randomized search runs one pipeline as multiple feature selection techniques cannot be implemented in one pipeline or one search. Furthermore, the scaler chosen to normalize the data is StandardScaler. The randomized searches are run with a 3-fold cross-validation scheme. The reason for choosing randomized search cross validation over grid search cross validation is that it has a complexity of less than half of grid search while producing almost the same results. A grid search cross validation scheme was implemented before but the time complexity of the program was significantly high. The results from the randomized searches are outputted such as the features selected and the best scores achieved.

The feature selection method used is SelectKBest which returns the top K number of features based on a statistical score; the statistical used to establish dependencies are Mutual Information and ANOVA-f. ANOVA-f is useful in the feature selection process when the input variables are numerical and the target variable is categorical; it establishes a linear relationship between the independent variables and the dependent (target) variable. Mutual Information is not limited by a single dependency technique (eg. Linear) while establishing correlations with the target variable, it can be used to find the importance of features by different statistical dependency methods which are relevant. The use of Mutual Information ensures that the features selected by just one statistical approach are not incorrect.

The performance metric used is the F1 score. Although the data is not imbalanced, checking for false positives and false negatives is of high importance in cybersecurity. In order to implement the model in a dynamic scenario, F1 score also defends against data drift wherein the dynamic data later used to train the model could be imbalanced. In such cases, the accuracy score would continue to be high while the model would not be able to correctly classify malignant DNS flows. Furthermore, the models chosen are also well-suited to scenarios like this.

The scores from the randomized search are compared and then estimator that achieved the highest score is used to create a model for the evaluation phase. In the evaluation phase, The training set and testing set created from the split earlier are normalized and only the list of features received from the top grid are used. A model is created using the best hyperparameters from the search to make predictions on the test data. Results show that the chosen model performs very well with a high rate of True Positives and an F1 score of about 0.86 (see fig. 4, fig. 5). Moreover, the features selected from the results also agree to the analysis made on the dataset initially using heatmaps and mutual information plots; this just confirms the observations. Although the performance of the models was similar, the top model is chosen for developing the static model. Upon multiple runs of the program, it is observed that the Gradient Boosting model along with feature selection based on Mutual Information performs slightly better than the others.

Final Model

Lastly, a static model is created from all the aforementioned techniques and searches so that it can be deployed in a dynamic environment. The entire dataset is scaled using the StandardScaler and the best features are selected. A model is once again created from the best estimator and hyperparameters which is fitted to the entire dataset. The created model is then saved as a pickle file to be loaded up as a dynamic model.

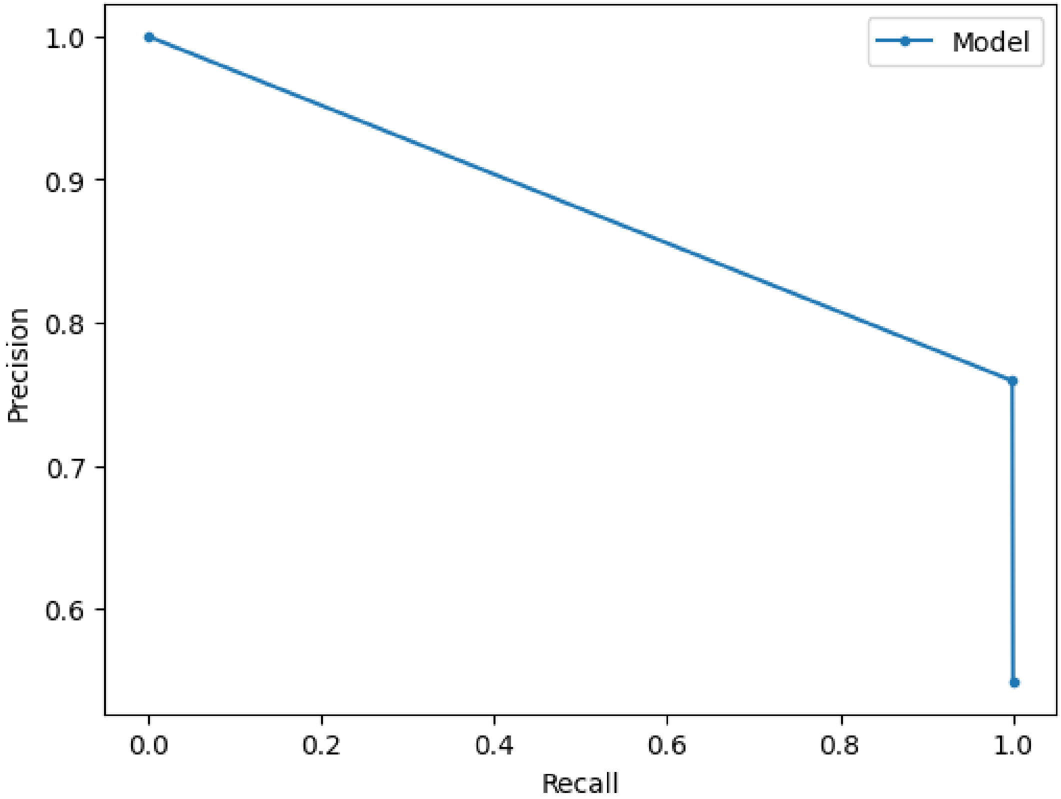
Other than the model, the scaler is also saved as a pickle file which can be used to transform the incoming data based on the parameters obtained by fitting it on the static dataset. Additionally, a Boolean list containing the features to be selected is also saved as a pickle file so that it can be used to drop unrequired features from incoming data in the dynamic environment.

Dynamic Model

In the dynamic implementation, the model, scaler, and features list are loaded from pickle files. Data is dynamically received in a for loop which is stored in lists until a count of 1000 is reached; upon reaching 1000, the batch is added to a dataframe which is used to test a static model and a dynamic model. The batch is also added to another dataframe which contains data from the previous batches as well. This main dataframe will be used to retrain the model if required. The retraining of the model occurs by setting the hyperparameter 'warm_start' to true for the model which allows additional estimators to be added which could fit the data.

The batches of 1000 are sent to a testing function where two models are evaluated using the F1 score. If the F1 score of the dynamic model falls below a threshold, it is retrained using all the new data accumulated. If the dynamic model was retrained, a Boolean value is returned which indicates the retraining process and so the main dataframe containing the batches is cleared and re-initialized so that the model is not retrained on the same data again. The static model is not retrained so it serves as a comparison of how retraining helps in maintaining a consistent score.

The threshold chosen for retraining is 0.85 because it is found that the score sometimes dips to approximately 0.83 where the model could be retrained to bolster the performance again.



The F1 scores over the batches indicate that retraining the model increases the performance (see fig. 6) and is beneficial as it protects against data drift in which the incoming data could be new or different from the data on which the static models were changed. This provides safety against evolving attacks.

Appendix

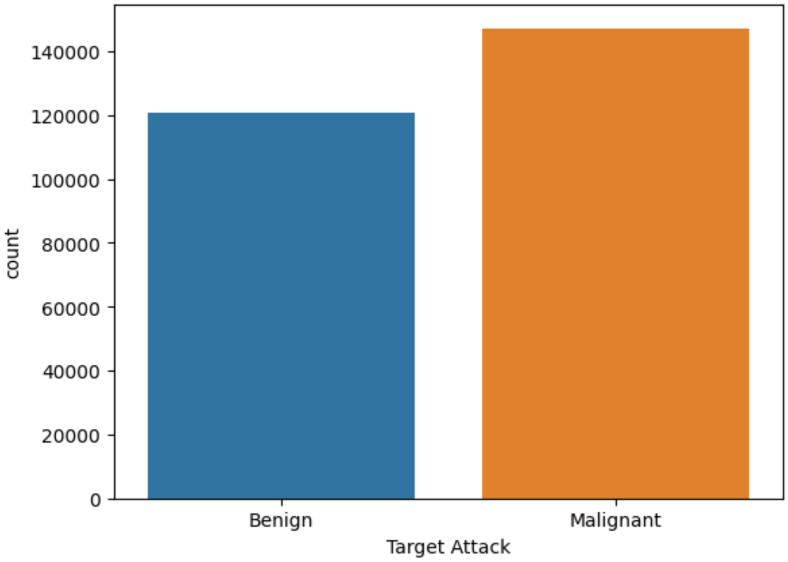


Figure 1 Value Counts of the set

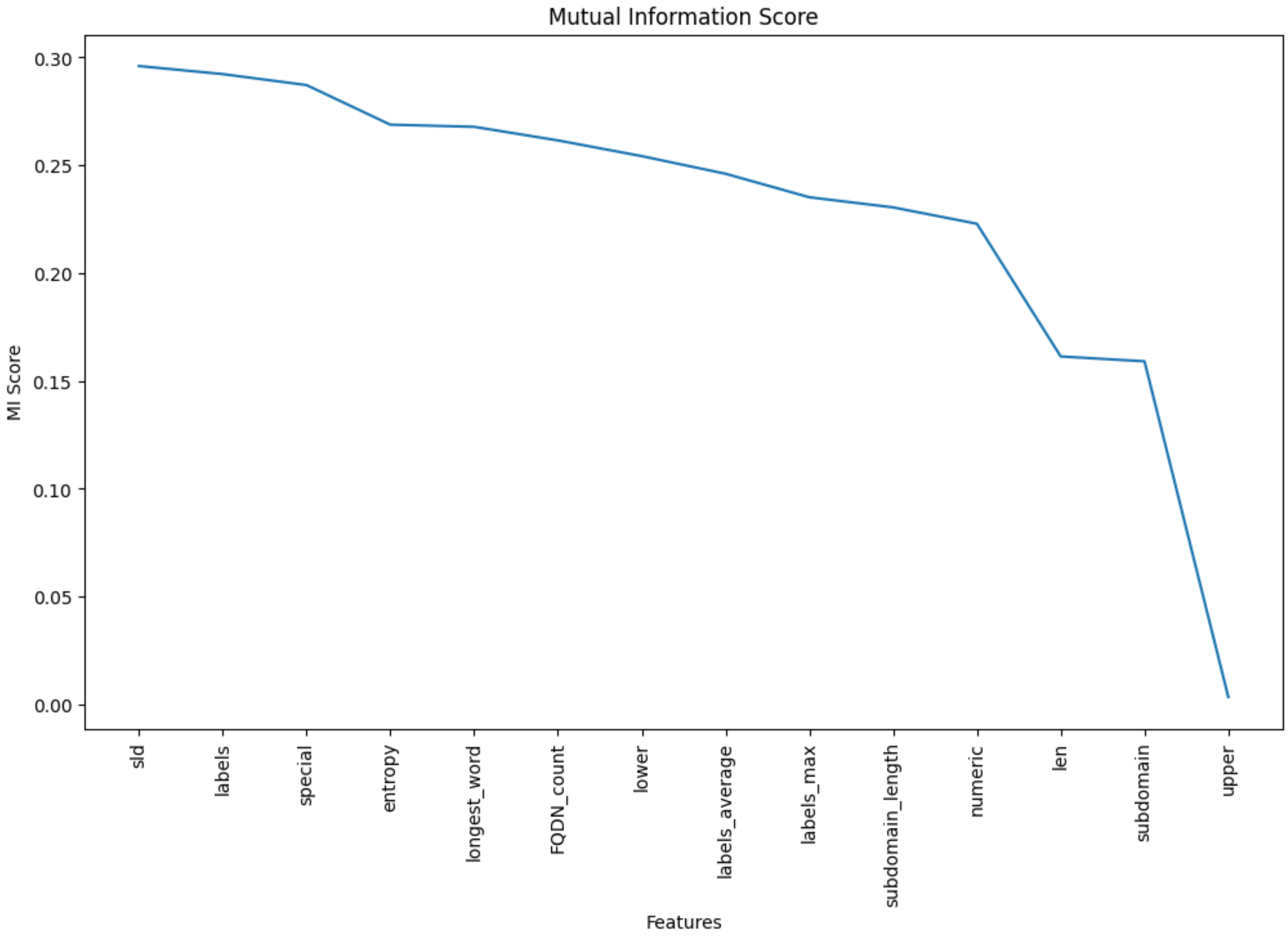


Figure 2 Mutual Information Importance

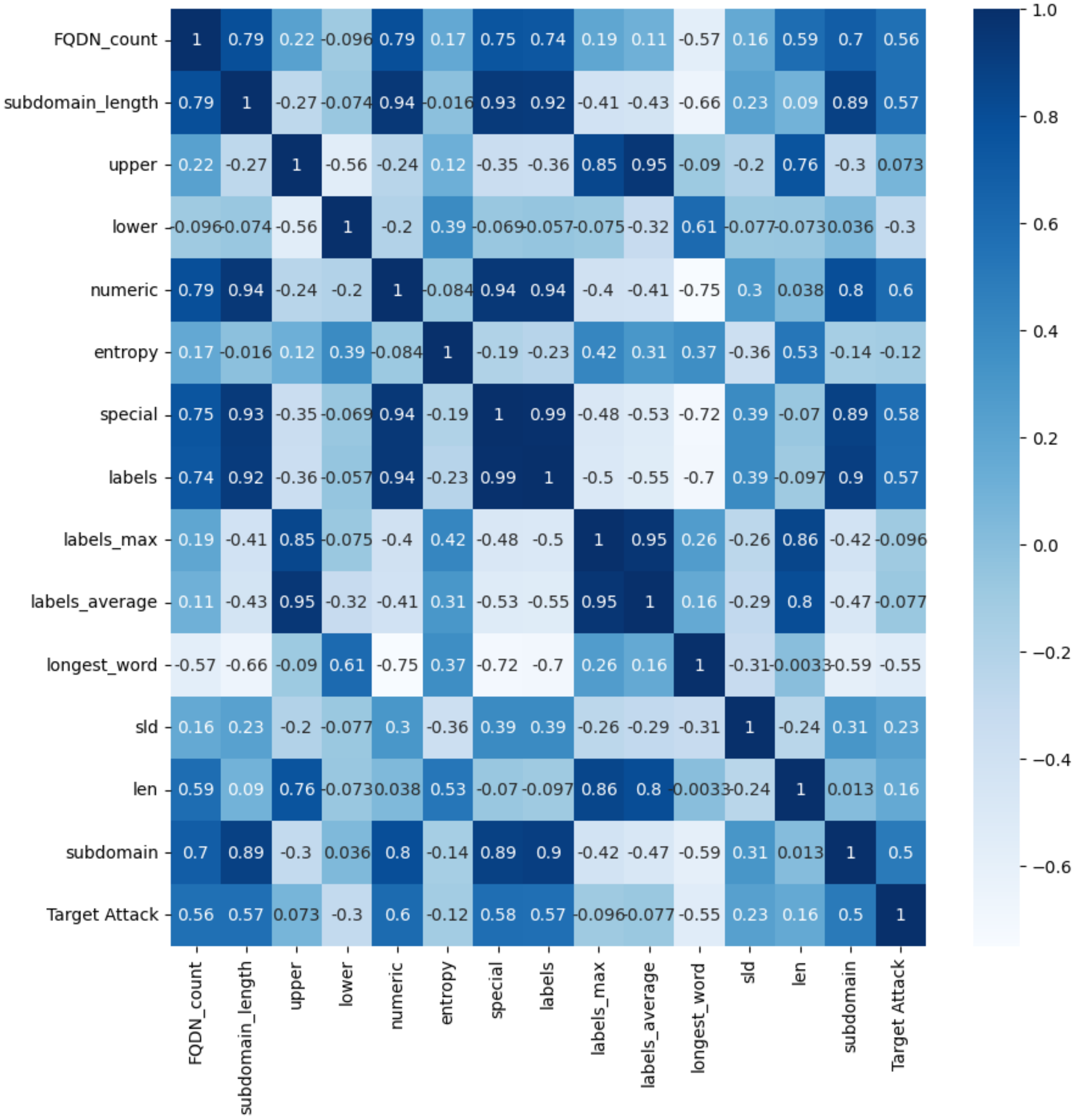


Figure 3 Correlation heatmap

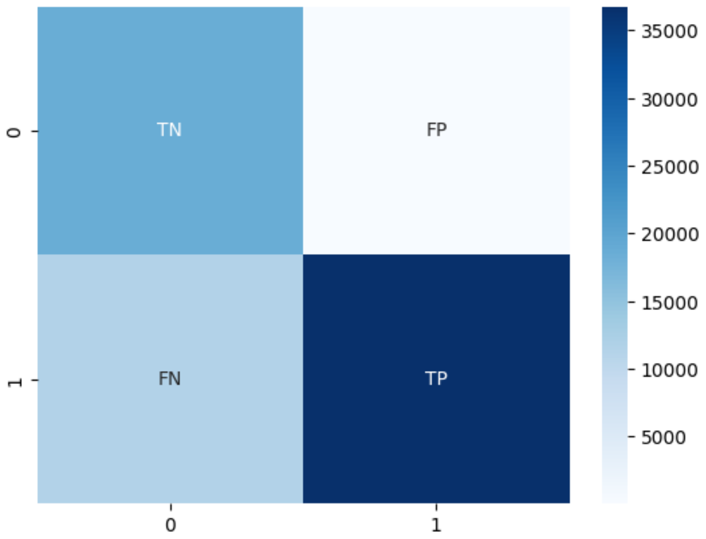


Figure 4 Confusion Matrix

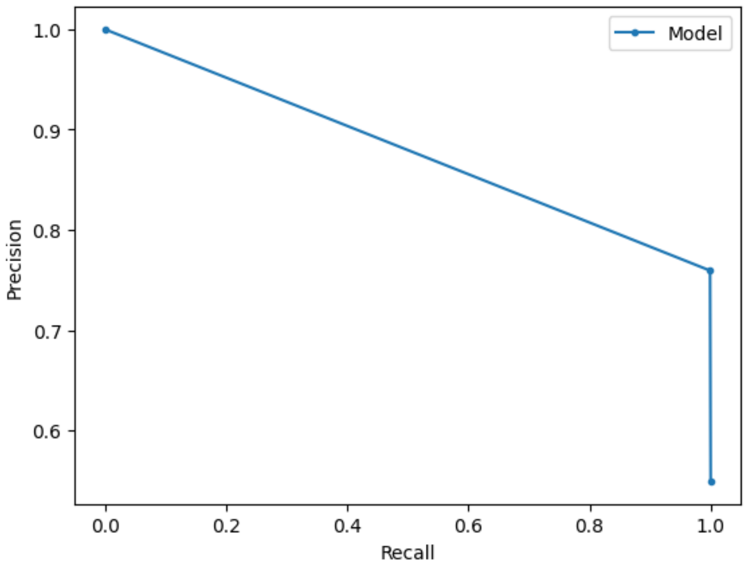


Figure 5 Precision-Recall Curve

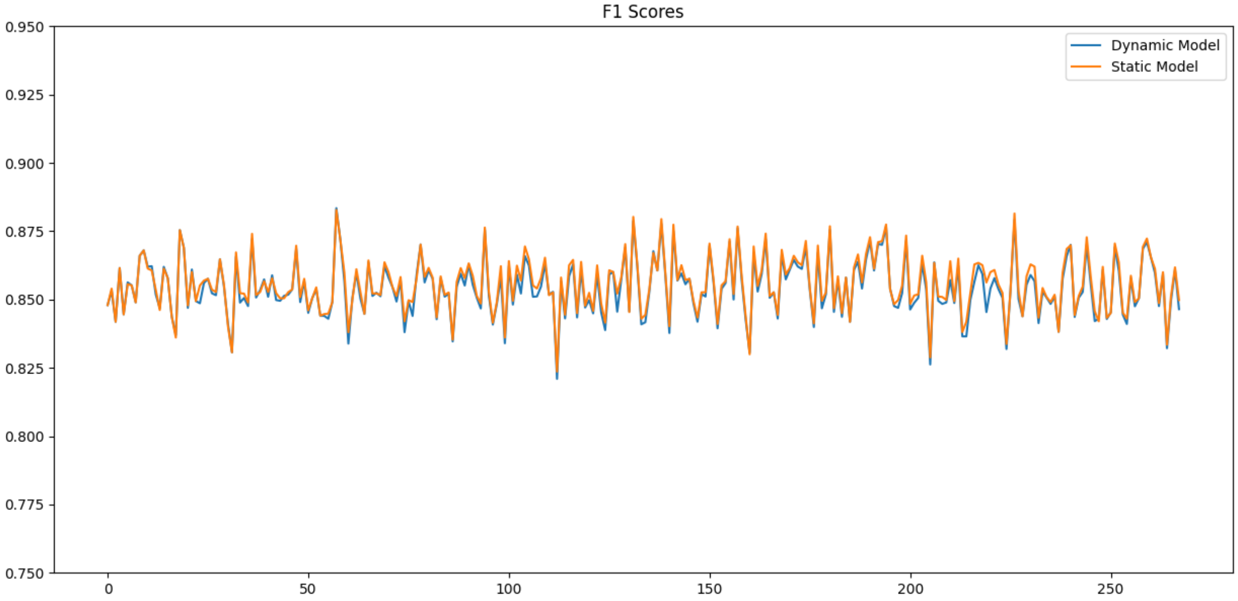

F1 Scores
Dynamic Model
Static Model
0.950
0.925
0.900
0.875
0.850
0.825
0.800
0.775
0.750
0
50
100
150
200
250

Figure 6