uOttawa

ELG7186 AI for Cyber Security
"Continuous Evaluation with Kafka"

Assignment 3

Instructor: Dr. Paula Branco

Student's Name: Ali Amin El-Sayed Mahmoud El-Sherif ID: 300327246

Table of Contents

Table of Figures

Part I

1.

The counts of each unique values in each column were calculated and the data distribution was presented as well, and here's an example for the feature 'upper' in fig (2). The target column had a predominantly balanced data distribution on classes 0 and 1 as shown in fig (3). The skewness level of each column was calculated as in fig (1), nevertheless, plotted for each column as in fig (4) for 'labels_average' and all of them were sorted in descending order from the highest level of skewness and classified into 3 levels: Highly skewed (more than 1 or less than -1), moderately skewed([-1, -0.5] & [0.5, 1]) and fairly skewed [-0.5, 0.5].

```
FQDN_count           -1.101731
subdomain_length     -0.590480
upper                 5.988737
lower                 0.343449
numeric              -0.594384
entropy              -0.140156
special              -0.902972
labels               -0.903680
labels_max            3.979910
labels_average        5.087081
longest_word          2.269378
sld                 180.987411
len                   2.634801
subdomain            -1.176397
Target Attack        -0.197046
dtype: float64
Number of skewed columns:  15
```

Figure 1 Skewness levels

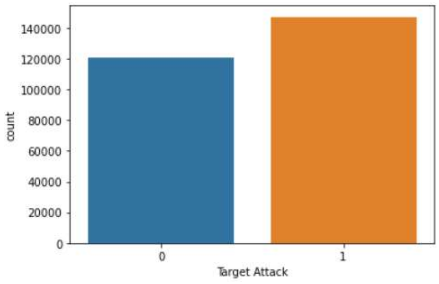

Figure 2 Target attack distribution

```
Skewness =  5.087081
<AxesSubplot:ylabel='Frequency'>
```

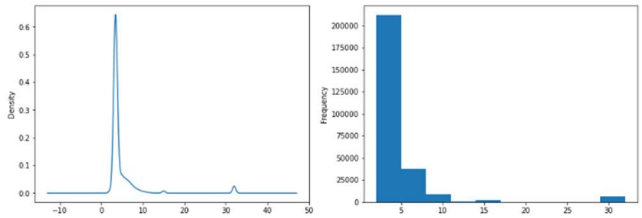

Figure 4 Skewness showing

```
0      258759
32       6345
11       1961
2        1008
8           1
Name: upper, dtype: int64
```

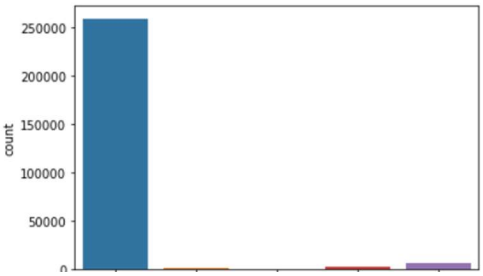

Figure 3 upper distribution

2.

By inspecting the data, it was shown that 'longest_word' and 'sld' are considered as type object with 8 missing values that were both handled through using regular expressions (regex) by replacing them and filling the null values with '2' and '192' as the most used values in each column, and 'timestamp' was dropped.

3.

In this part, a heatmap was used to show correlations between features in fig(5), "Target Attack" is highly correlated with: "FQDN_count","subdomain_length","numeric","special","labels" and "subdomain". "labels max" is highly correlated with: "labels_average". "subdomain_length" is highly correlated with: "subdomain","special","labels" and "numeric". "numeric" is highly correlated with: "subdomain_length","special" and "labels". "len" is highly correlated with: "labels_max". "subdomain" is highly correlated with: "labels","special" and "subdomain_length". "longest_word" is highly correlated with: None. "sld" is highly correlated with: None. "upper" is highly correlated with: "labels_max","labels_average" and "len".

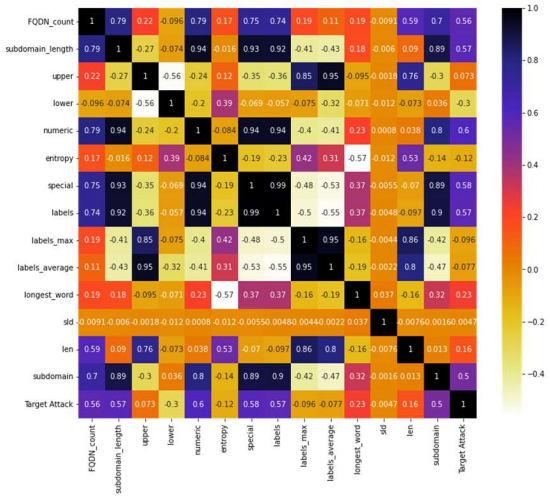

Figure 5 Heat map

Also, mutual information technique was used to show the importance of features as in fig (6) and fig (7). followed by ANOVA to add credibility to it, which resulted the following order: 'FQDN_count', 'subdomain_length', 'upper', 'lower', 'numeric','entropy', 'special', 'labels', 'labels_max', 'labels_average','longest_word', 'sld', 'len', 'subdomain' as in fig(8).

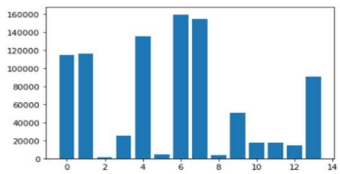

Figure 8 ANOVA feature importance with indices

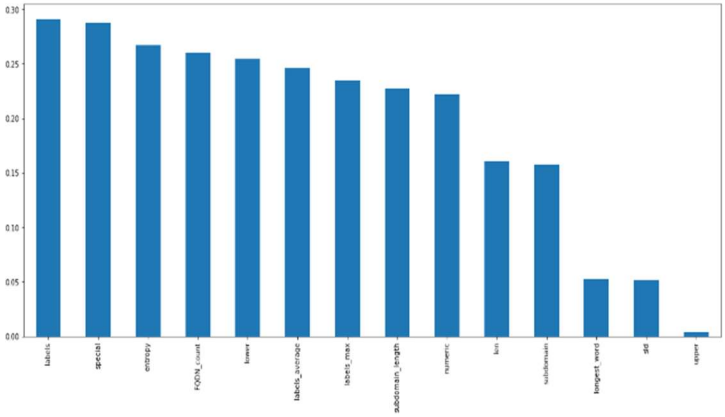

Figure 6 Mutual information feature importance

labels	0.290694
special	0.287702
entropy	0.266832
FQDN_count	0.260013
lower	0.254588
labels_average	0.246338
labels_max	0.235035
subdomain_length	0.226925
numeric	0.222268
len	0.160279
subdomain	0.157714
longest_word	0.052162
sld	0.051716
upper	0.003778

Figure 7 Mutual

Based on the highest scores in heatmap, mutual information and ANOVA, the list of columns is:

'FQDN_count', 'subdomain_length', 'upper', 'lower', 'numeric','entropy', 'special', 'labels', 'labels_max', 'labels_average', 'len'.

4.

The data was shuffled using ".sample(frac=1)" then splitted using "floor" and "ceil". MinMax scaler was used, then PowerTransformer for normalization using "yeo-johnson" method to deal with negative values as boxcox cannot. A comparison of output data was compared with the original one in each column by skewness level, KDE plot and histogram as shown in the example of the column "labels_average" in fig (9):

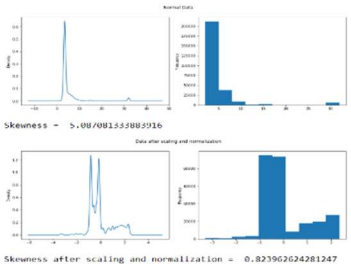

Figure 9 Effects of scaling and normalization

XGboost (fig 12,11) and logistic regression(fig 10,13) were chosen with the following outputs:

Train accuracy : 82.67%
Test accuracy : 82.35%

	precision	recall	f1-score	support
0	1.00	0.61	0.76	24183
1	0.76	1.00	0.86	29432
accuracy			0.82	53615
macro avg	0.88	0.80	0.81	53615
weighted avg	0.87	0.82	0.81	53615

Figure 13 Classification report of XGboost

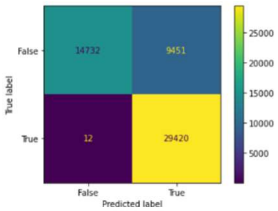

Figure 12 Confusion matrix of XGboost

Train accuracy : 82.54%
Test accuracy : 82.22%

	precision	recall	f1-score	support
0	0.99	0.61	0.76	24183
1	0.76	1.00	0.86	29432
accuracy			0.82	53615
macro avg	0.88	0.80	0.81	53615
weighted avg	0.86	0.82	0.81	53615

Figure 11 Classification report of LR

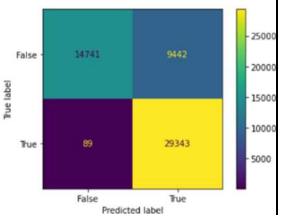

Figure 10 Confusion matrix of LR

XGboost was picked as the model better:

Train accuracy : 82.67%
Test accuracy : 82.35%

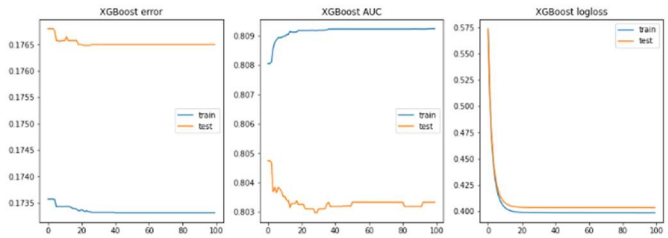

Figure 14 XGboost evaluation

Hyperparameter tuning was done on XG boost using randomized search resulting fig (15):

The baseline XGboost was slightly better than the tuned one by 0.01% in both training and testing, which is the champion model.

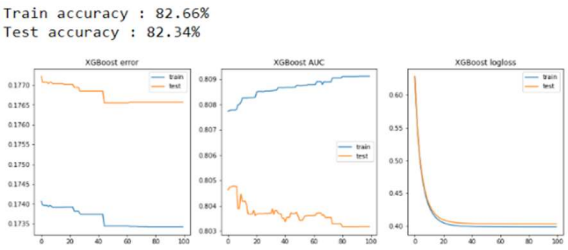

Figure 15 Hyper parameter tuned XGboost

5.

10 fold cross validation was used on the champion model regarding f1-score, recall, accuracy and precision, and, it was plotted in the following fig(16):

Since the metrics were close, this means that the model was not preferring a certain metric and neglecting the other which is a good trait.

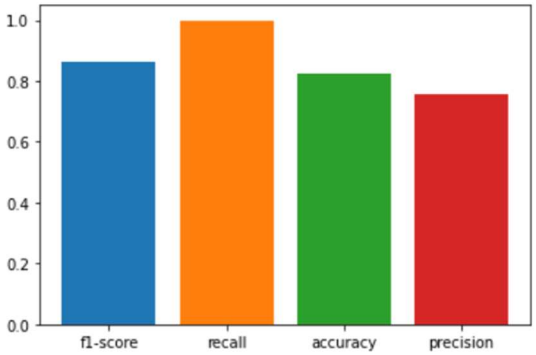

Figure 16 Cross validation comparison with different metrics

Part II

1.

Kafka and dockers were installed using dependencies based on the provided PDF.

2.

"Kafka_dataset.csv" was used based on instructions.

3.

The chosen model which was saved as .pkl file was loaded as requested.

4.

Two models were created through different functions including data cleaning procedures, iterations, data loading and more.

5.

1000 streaming data is used in each period, these 1000 are predicted by dynamic model and evaluated through specific threshold, if the performance is less than 82.5% the model is retrained and store the new metric score for evaluation later on.

6.

The consumer code was run through 262 window and here is the final one in fig(17):

```
Window number: 262
Accuracy of dynamic model = 82.69999999999999%
Accuracy of static model = 83.2%
********************
```

Figure 17 last window ran

7.

In fig (18), a plot showing the accuracy differences between static and dynamic models. A decision boundary was made to retrain the model at any value less than 82.5% f1-score accuracy.

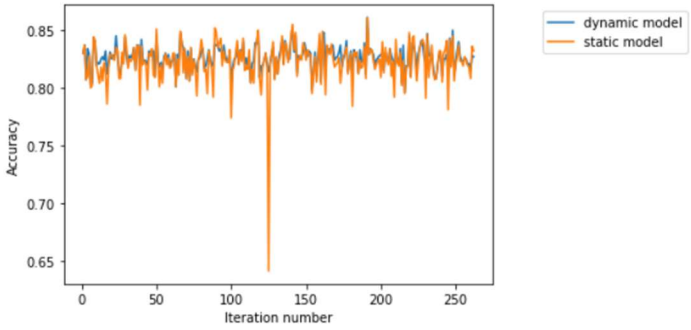

Figure 18 Dynamic and static models comparison

8.

Evaluation was done on each model on every single window as in fig(17).

9.

Two arrays were created for dynamic and static models to store performance metrics. A comparing plot was made fig (18).

10.

82.5% threshold was chosen as the score of the static model (XGboost) was 82.67% on training and 82.35 on testing, so it seemed a reasonable point to be the least to accept.

11.

It was obvious that both of models were so close in performance which means that the static model is well generalized. There was not a lot of challenge for the model because the new data distribution was not so different from the original data.