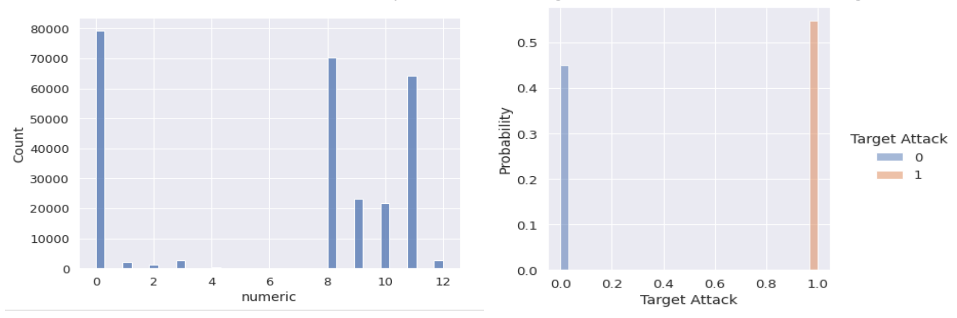
ELG7186 – AI for Cybersecurity Applications

Assignment 3

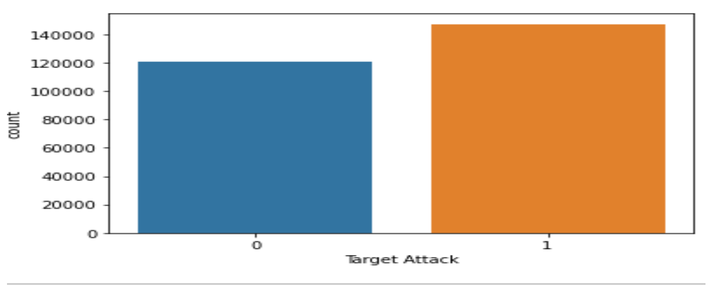
Part I (Static Model):

1. **Data Analysis**: First, I loaded the Static Dataset then I created a basic histogram for every feature. to divide the value range of continuous variables into discrete bins and show how many values exist in each bin, for example, this histogram for numeric and target

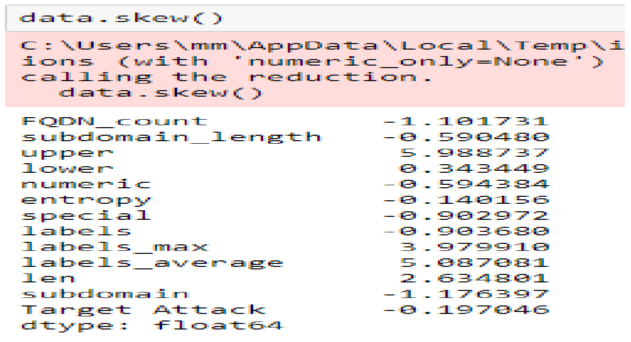


The dataset is balanced somehow as we show in this figure, and my interest in class one.

```
AxesSubplot:xlabel='Target Attack', ylabel='count'>
```



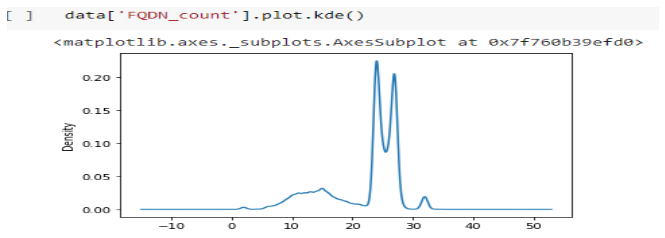
There is a type of data skewed pattern



```
data.skew()
C:\Users\mm\AppData\Local\Temp\i
ions (with 'numeric_only=None')
calling the reduction.
   data.skew()
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
len                      2.634801
subdomain               -1.176397
Target Attack           -0.197046
dtype: float64
```

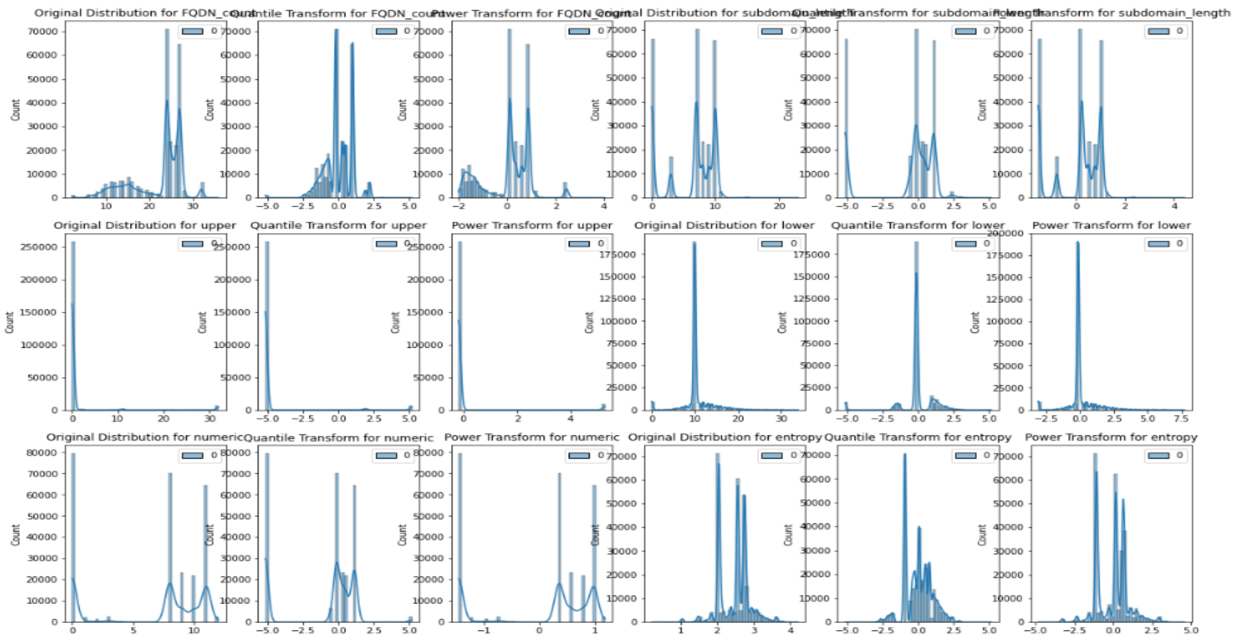
I showed this in plots and this plot shows the “FQDN_count “feature and made it for other features.

The models are decision trees and Adaboost it doesn’t need data to be a normal distribution.

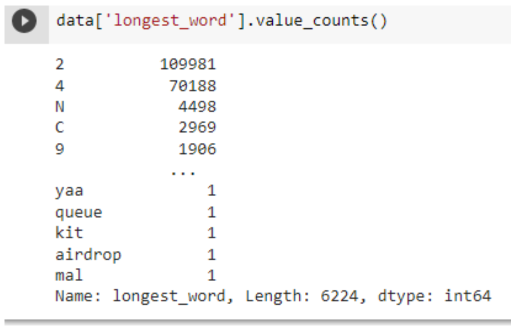


```
[ ]   data['FQDN_count'].plot.kde()

<matplotlib.axes._subplots.AxesSubplot at 0x7f760b39efd0>
```

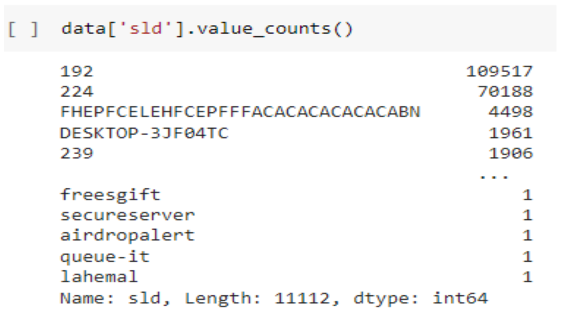
And this is the distribution of all features in the dataset, there are negative and positive skew and the models are decision trees, and Adaboost doesn't need data to be a normal distribution.



1. **Feature engineering and data cleaning:** firstly, transform the variables that contain string values, there are strings in the "longest word" and "sld" features

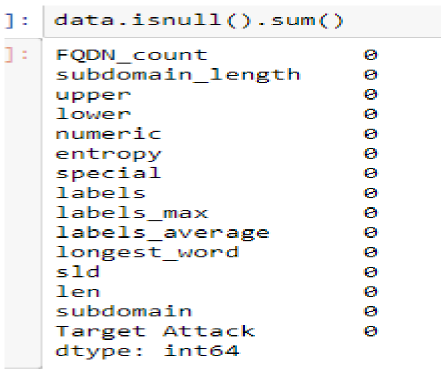


```
data['longest_word'].value_counts()

2           109981
4            70188
N             4498
C             2969
9             1906
             ...
yaa              1
queue            1
kit              1
airdrop          1
mal              1
Name: longest_word, Length: 6224, dtype: int64
```



```
[ ]  data['sld'].value_counts()

192                                       109517
224                                        70188
FHEPFCELEHFCEPFFFACACACACACACABN            4498
DESKTOP-3JF04TC                             1961
239                                         1906
                                           ...
freesgift                                      1
secureserver                                   1
airdropalert                                   1
queue-it                                       1
lahemal                                        1
Name: sld, Length: 11112, dtype: int64
```

And solving this by replacing the strings values with the most present values like (2) in "longest word and (192) in "sld". Then checked the missing values there are 8 non-values in 'longest word' and I handled them.

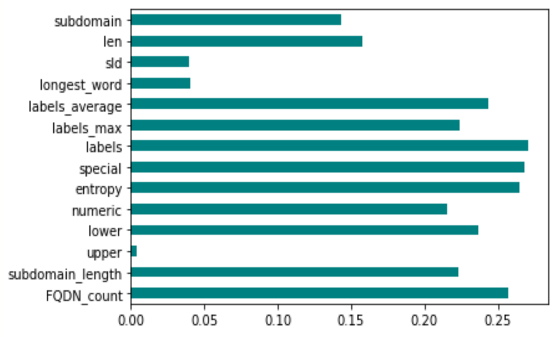


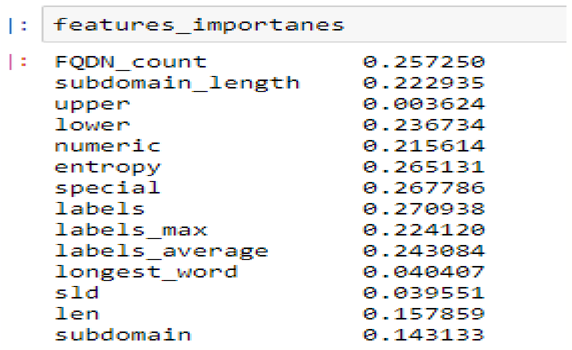
```
]: data.isnull().sum()

]: FQDN_count           0
   subdomain_length     0
   upper                0
   lower                0
   numeric              0
   entropy              0
   special              0
   labels               0
   labels_max           0
   labels_average       0
   longest_word         0
   sld                  0
   len                  0
   subdomain            0
   Target Attack        0
   dtype: int64
```

after handled them as we show there is no missing values

3-Feature Filtering/Selection:

Firstly, I split the data into (training, validation, and testing) I choose this until data leakage doesn't happen, to apply feature selection on the validation set and train my models in the training set finally testing the models using the test set.
I have used the validation set in feature selection and Hyperparameter tuning to decide which parameters are the best for each model
I have used the "Chi_Squre", "Mutual Info", "And ANOVA" as different statistical techniques to evaluate which features are the most importance
to train the model
Chi-square gives the most important features 'FQDN_count', 'subdomain_length', 'numeric', 'special', and 'labels'.
The **ANOVA** technique gives me 'FQDN_count', 'subdomain_length', 'numeric', 'special', and 'labels'.
And **Mutual Info** gives the 'labels', 'special', 'entropy' the most important features.
This graph shows the importance of features after applying Mutual Info.





```
]: features_importanes

]: FQDN_count          0.257250
   subdomain_length    0.222935
   upper               0.003624
   lower               0.236734
   numeric             0.215614
   entropy             0.265131
   special             0.267786
   labels              0.270938
   labels_max          0.224120
   labels_average      0.243084
   longest_word        0.040407
   sld                 0.039551
   len                 0.157859
   subdomain           0.143133
```

All this technique agreed to choose the 'subdomain length' and 'labels' and this is normal because these features are the most important when classifying the attack. After Applying the feature Selection techniques then I Normalized these features to train the models.

4- Model Training

On the other hand, I applied hyperparameter tuning using grid search on the models that I selected them using the validation set until data leakage doesn't happen. I chose the Logistic Regression, Decision tree, and AdaBoost classifier as models for training, For each of these models, I have trained three times, one on the result after applying chi-Square, another on the result after Applying ANOVA, and the last one after applying mutual Info.

The target is predicting all attacks, which means a high recall so a low false positive threshold so the performance evaluation matrix will be the best is **recall.**

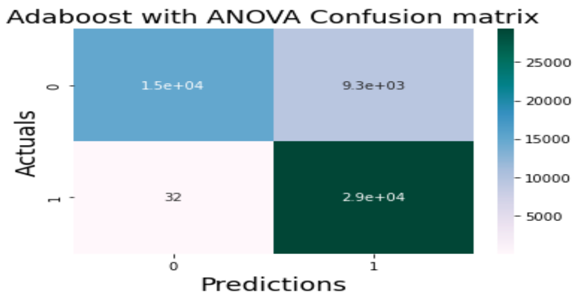
This table shows the accuracy, recall, and precision.

	Chi_LR	Chi_DT	Chi_adaboost	mutual_ LR	Mutual_ DT	Mutual_adaboo st	ANOVA _LR	ANOVA_ DT	ANOVA_adaboost
Accura cy	0.80	0.83	0.83	0.80	0.82	0.83	0.80	0.83	0.83
Recall 0	0.63	0.62	0.62	0.63	0.61	0.61	0.63	0.61	0.61
Recall 1	0.94	1.0	1.0	0.95	1.0	1.0	0.94	1.0	1.0
Precisio n 0	0.90	1.0	1.0	0.91	1.0	1.0	0.90	1.0	1.0
Precisio n 1	0.76	0.76	0.76	0.75	0.76	0.76	0.76	0.76	0.76

5- Model evaluation:

I have selected (The AdaBoost classifier with (n_estimators=500, random_state=0 , learning rate=1.0) as the best parameters with ANOVA as feature selection) as the best model that gives accuracy =83 % and class one recall's = 1.0, so this is the champion model . this is the confusion matrix and classification report of the best model.

```
Confusion Matrix:

[[14959  9314]
 [   32 29310]]
```



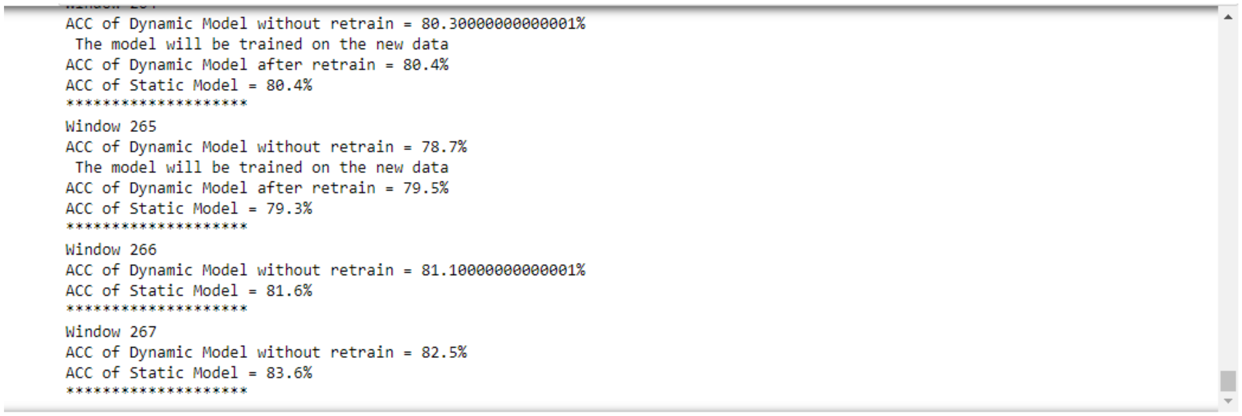
```
              precision    recall  f1-score   support

           0       1.00      0.62      0.76     24273
           1       0.76      1.00      0.86     29342

    accuracy                           0.83     53615
   macro avg       0.88      0.81      0.81     53615
weighted avg       0.87      0.83      0.82     53615
```

Then I applied the cross validation on the champion model. And these are the cross-validation scores

```
Cross Validation Scores:  [0.82439616 0.82598154 0.82411639 0.82389878 0.82563417]
Average CV Score:  0.8248054066596451
Number of CV Scores used in Average:  5
```

Then I saved the champion model as a static model to use in the second part.

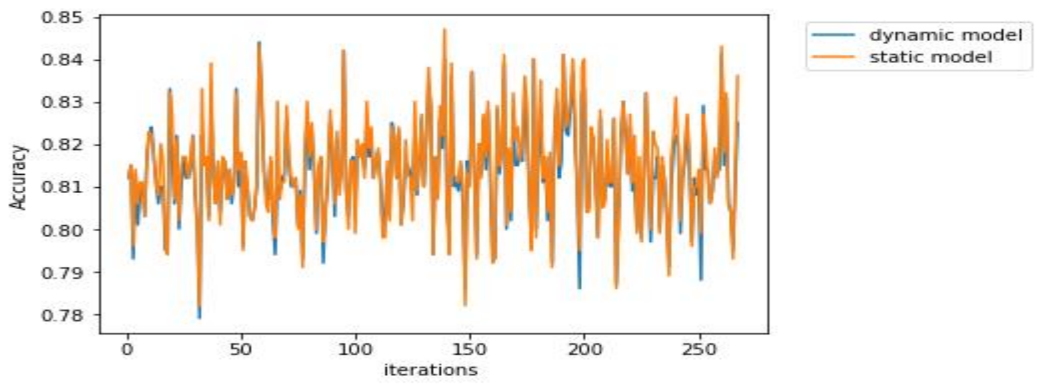
The Second Part: firstly, I follow up on install instructions to setup docker and create the images, then I used the Kafka dataset in the producer. In consumer, I loaded the static model that I saved to use it. Firstly, I get 1000 records as data streaming information and use them as a window and I applied data preparation on these records to convert them to a data frame. then I transform the variables that contain string values and remove missing values .split the data into X and Y then loaded these records into static data set, and made a prediction on these records if accuracy is less than 0.81 because of this is the accuracy of the static model that I saved, will retrain the model with passing the data after applying the ANOVA feature selection and train the AdaBoost classifier on training data then will check the accuracy and evaluate the dynamic model's performance and store the accuracies to evaluate the performance of both models. This figure show us the latest windows



```
ACC of Dynamic Model without retrain = 80.30000000000001%
 The model will be trained on the new data
ACC of Dynamic Model after retrain = 80.4%
ACC of Static Model = 80.4%
********************
Window 265
ACC of Dynamic Model without retrain = 78.7%
 The model will be trained on the new data
ACC of Dynamic Model after retrain = 79.5%
ACC of Static Model = 79.3%
********************
Window 266
ACC of Dynamic Model without retrain = 81.10000000000001%
ACC of Static Model = 81.6%
********************
Window 267
ACC of Dynamic Model without retrain = 82.5%
ACC of Static Model = 83.6%
********************
```

And this is a time-based plot comparing both performances from both models.

```
37]: <matplotlib.legend.Legend at 0x2de662f0b50>
```



There is a slight improvement over the dynamic model because the data come from the same distribution, if there is real data it will be a big improvement .it is important to retrain our model with there is new data comes and new features.
The limitations: we need high computational memory as we append 1000 records every window. Over time we will need more memory but there is a benefit because we store the historical attacks and new attacks to the model can detect both of them.
Overall from this assignment firstly I have learned when I train the models I should use data not used in feature selection and hyperparameter tuning so that it does not happen data leakage. secondly it is essential to retrain the model if we have new data came