ELG7186 – AI for Cybersecurity Applications

Assignment 3
Continuous Evaluation with Kafka

Part I (Static Model)

1. Data Analysis

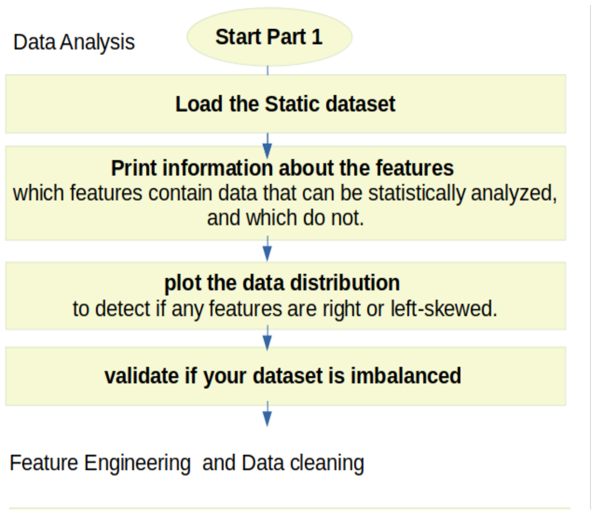

Data Analysis
Start Part 1
Load the Static dataset
Print information about the features
which features contain data that can be statistically analyzed, and which do not.
plot the data distribution
to detect if any features are right or left-skewed.
validate if your dataset is imbalanced
Feature Engineering and Data cleaning

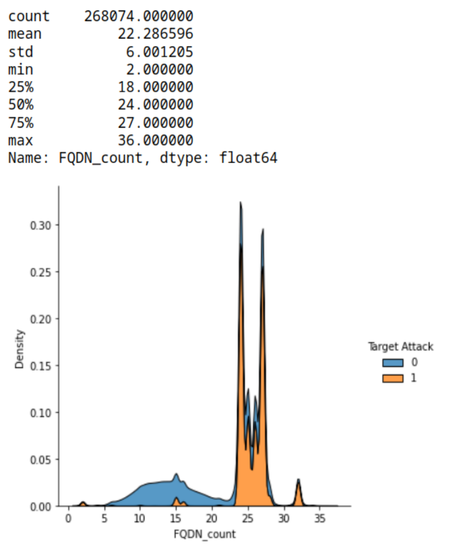
Table 1: summary of Data Analysis part

The static dataset consists of 15 features and targets.12 Features can be represented using the distribution plot, 3 Cannot until it is handled in the feature engineering phase.
A few features can be used, as its distribution has a clear space for a specific label.

	FQDN_count	subdomain_length	upper	lower	numeric	entropy
count	268074.000000	268074.000000	268074.000000	268074.000000	268074.000000	268074.000000
mean	22.286596	6.059021	0.845420	10.410014	6.497586	2.485735
std	6.001205	3.899505	4.941929	3.207725	4.499866	0.407709
min	2.000000	0.000000	0.000000	0.000000	0.000000	0.219195
25%	18.000000	3.000000	0.000000	10.000000	0.000000	2.054029
50%	24.000000	7.000000	0.000000	10.000000	8.000000	2.570417
75%	27.000000	10.000000	0.000000	10.000000	10.000000	2.767195
max	36.000000	23.000000	32.000000	34.000000	12.000000	4.216847

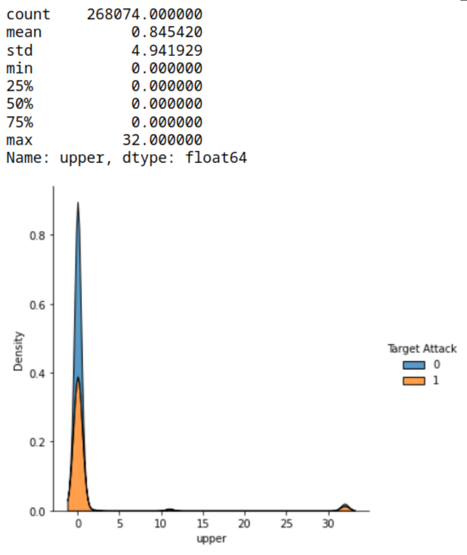
Table 2: example of features description

Check the distribution of the features



```
count    268074.000000
mean         22.286596
std           6.001205
min           2.000000
25%          18.000000
50%          24.000000
75%          27.000000
max          36.000000
Name: FQDN_count, dtype: float64
```

Figure 1 : right-skewed feature

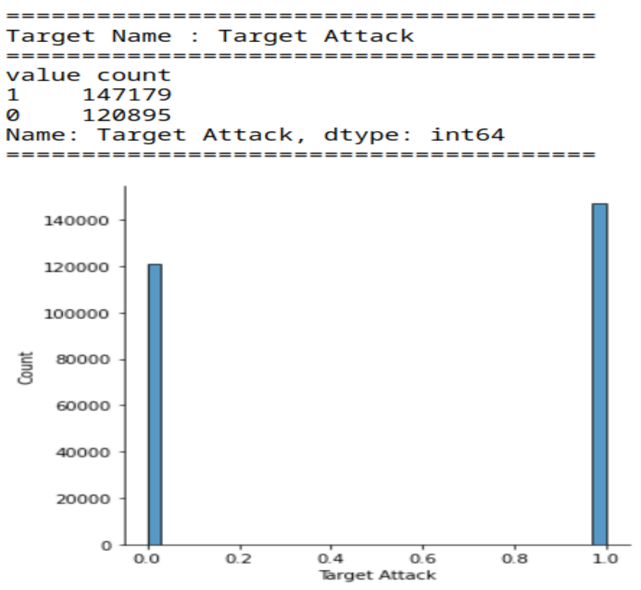


```
count    268074.000000
mean          0.845420
std           4.941929
min           0.000000
25%           0.000000
50%           0.000000
75%           0.000000
max          32.000000
Name: upper, dtype: float64
```

Figure 2: left- skewed feature

Note: for the rest of the graphs, please check the notebook for part one.

check the balance of the data

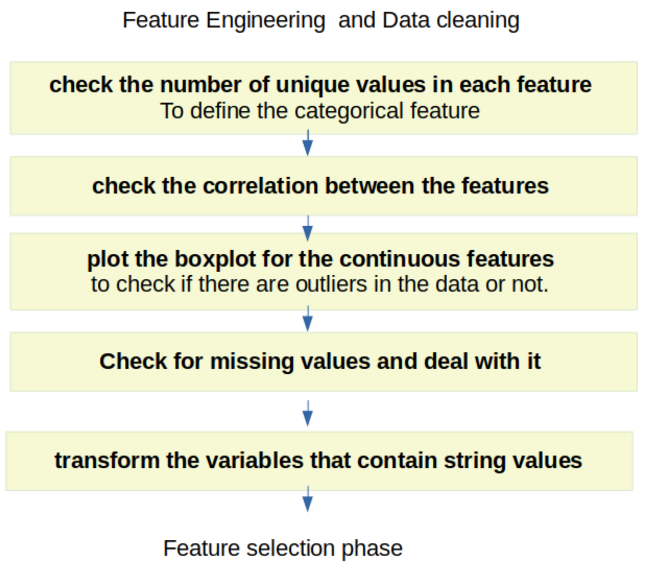


```
========================================
Target Name : Target Attack
========================================
value count
1     147179
0     120895
Name: Target Attack, dtype: int64
========================================
```

Figure 3: check the balance of the data

The data is balanced since we are interested in the positive class.

2. Feature engineering and data cleaning



Feature Engineering and Data cleaning
check the number of unique values in each feature
To define the categorical feature
check the correlation between the features
plot the boxplot for the continuous features
to check if there are outliers in the data or not.
Check for missing values and deal with it
transform the variables that contain string values
Feature selection phase

Table 3: summary of feature engineering phase

Some of the features contain categorical features since they have a small number of unique values. for example, “upper" represents the count of uppercase characters, "Labels” represents the number of labels in the domain," subdomain" represents whether the domain has a subdomain or not, and "special" represents the number of special characters. some of these features are logical to be categorical like subdomain. The other is not logical to consider as a categorical feature.

The rest of the features it has a large number of unique values, so I could not consider them categorical features.

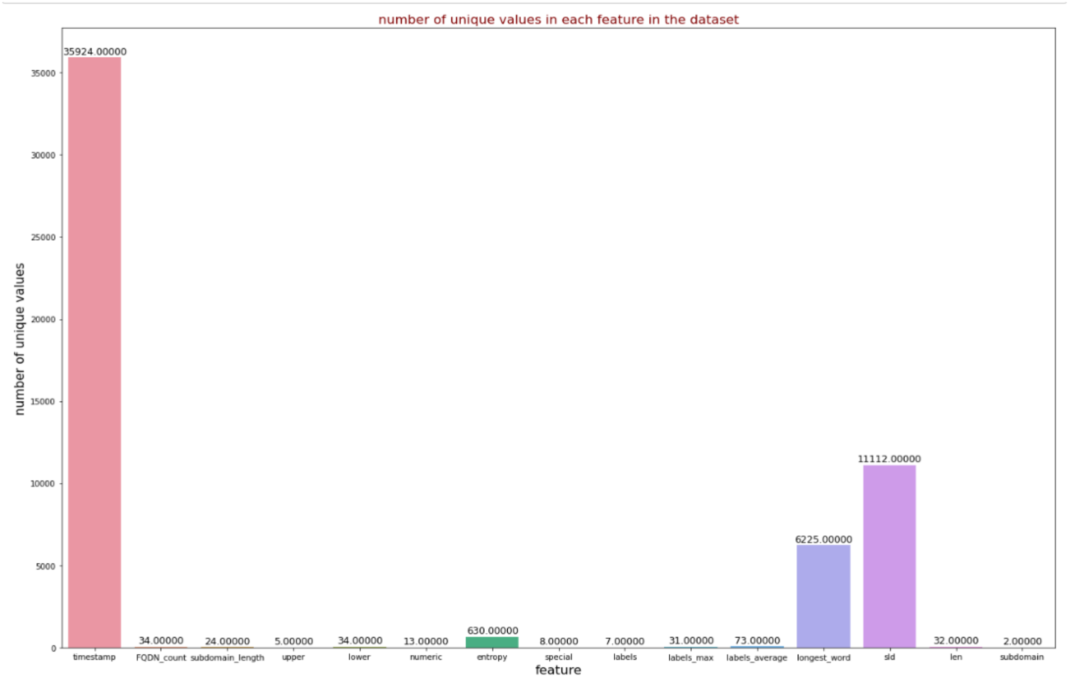


Figure 4 : unique values

correlation matrix

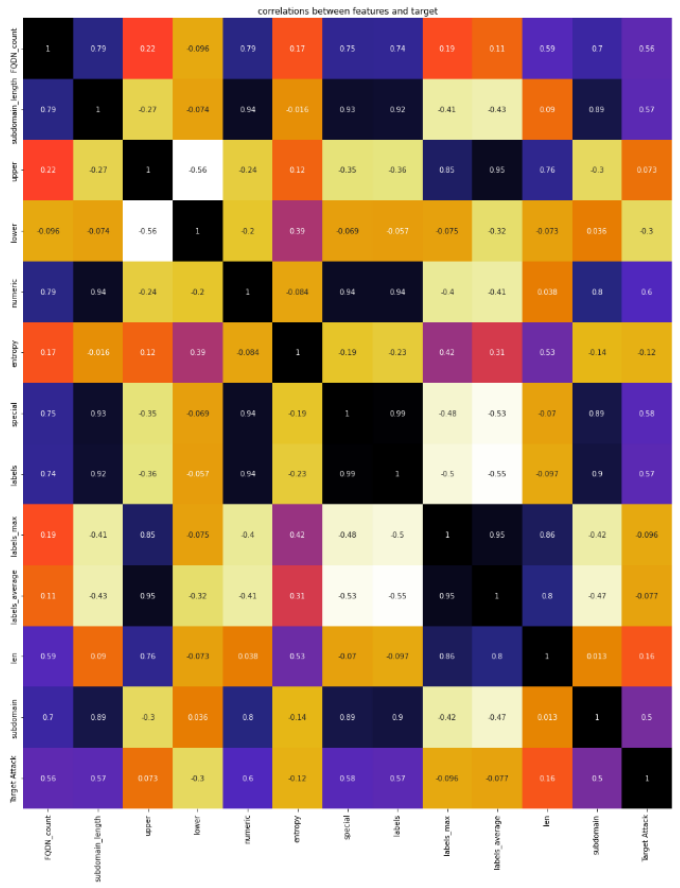


Figure 5: correlation matrix

The correlation matrix shows the correlations between the different features since each cell shows the correlations between two variables.

As the figure shows some features are highly positively correlated for example "subdomain" and "subdomain_length", but there are some features that are negatively correlated for example "upper" and "lower".

box plot

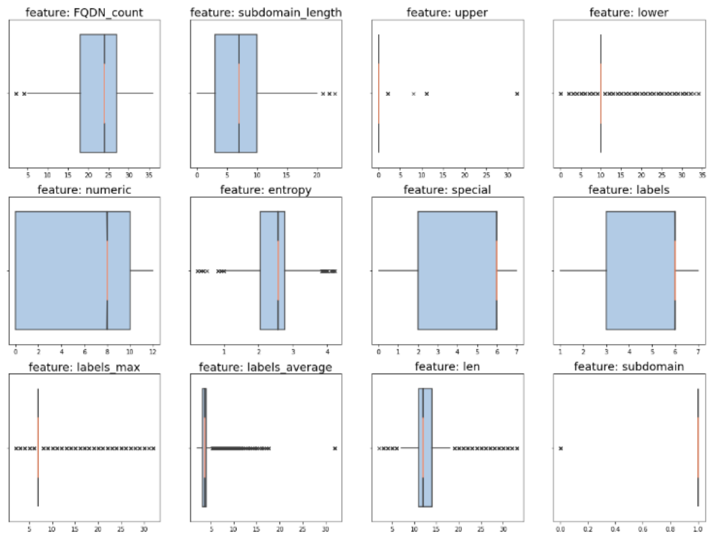
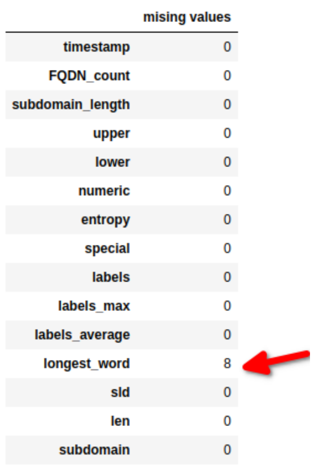


Figure 6: box plot for each feature

I have plotted the box plot for each feature so I can review the outliers in each feature since the outlier can be defined as a data point that is located outside the whiskers of the box plot.

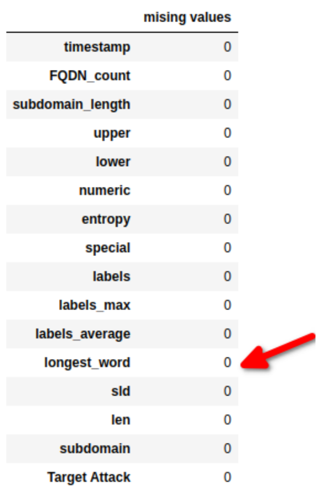
As the figure shows, some of the features have a lot of outliers. and the others do not.

Check for missing values and deal with it



	missing values
timestamp	0
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	8
sld	0
len	0
subdomain	0

Figure 7: check the missing values



	missing values
timestamp	0
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

Figure 8: check the missing values after dropping the 8 rows

I have checked the missing values in the data and found 8 missing values. I have decided to drop them since 8 values are few rows which may not affect the performance of the model in the next steps.

transform the features that contain string values

There are three features that contain String values: (“timestamp”, “longest_word”, and “sld”).

I decided to drop the timestamp column because all the other features in the dataset are stateless which are independent of the time-series characteristic of query domains and can be driven from individual query packets. So, none of the other features will be correlated with the timestamp, which makes me consider that it would have no effect on the classification decision.

The feature "longest_word" represents the longest meaningful word over the domain length. I thought about different ways to represent the longest word without dealing with it as garbage data. So, I have changed the longest word with the length to the word itself.
The figure shows the distribution of the values after dealing with it. As shown in the figure it has a high impact on identifying the target class.

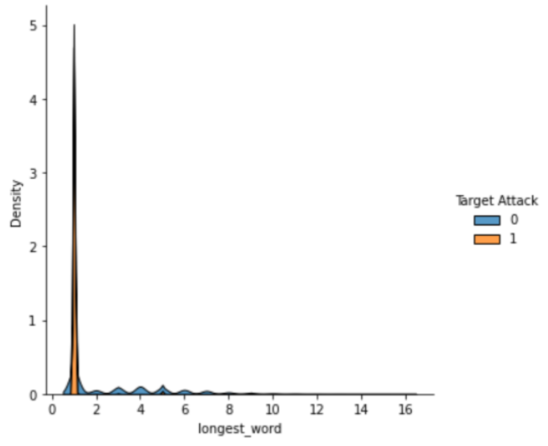


Figure 9: The distribution of the "longest_word" feature after dealing with the string values

The "sld" feature represents the second-level domain. I have also decided to change the value with the length to the word itself.

The figure shows the distribution of the values after dealing with it. As shown in the figure it has a high impact on identifying the target class.

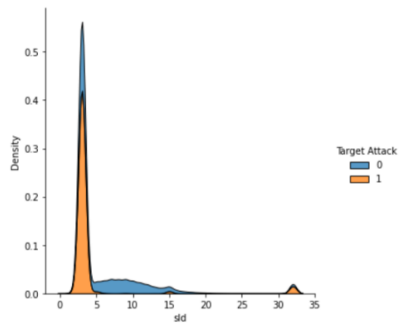
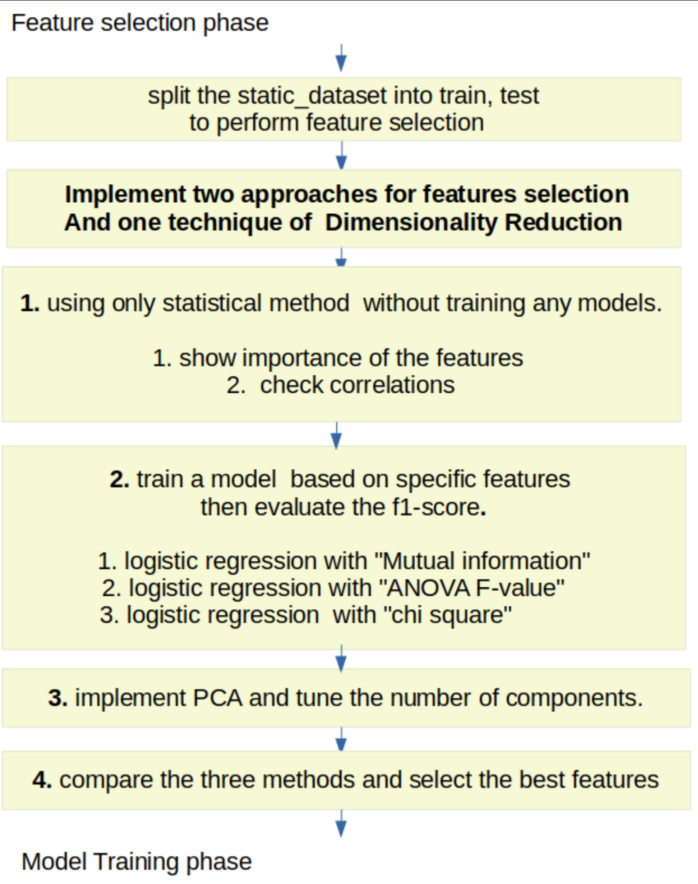


Figure 10: The distribution of the values in "sld" feature after dealing with the string values

3. Feature Filtering/Selection


Feature selection phase
split the static_dataset into train, test to perform feature selection
Implement two approaches for features selection And one technique of Dimensionality Reduction
1. using only statistical method without training any models.
1. show importance of the features
2. check correlations
2. train a model based on specific features then evaluate the f1-score.
1. logistic regression with "Mutual information"
2. logistic regression with "ANOVA F-value"
3. logistic regression with "chi square"
3. implement PCA and tune the number of components.
4. compare the three methods and select the best features
Model Training phase
Figure 11: feature selection phase

I have decided to split the data as a train and test. in the entire work I have used only the train to train and to check the performance of any model I have developed by using the cross-validation technique. The test set has been used twice in the entire work. first in the feature selection phase. Second, in testing the final model that has been chosen to be used in the next phase. I have not split the data as a train, validation, and test instead of train and test because I preferred to use the cross-validation technique with a large amount of the data so the model can be stable.
I have implemented two ways for feature selection and one way for dimensionality reduction first I have used only the Statistical test to select the feature by checking the importance of the features and the correlations between the features but, I have not chosen the features only based on this representation. So, I have trained a model in an iterative manner each time with a different combination of features and returned the f1-score as an evaluation metric for the model with these features.
I also have performed a dimensionality reduction task by using PCA and tuning the number of components to be used.
Finally, I have compared the different results and chosen the best set of features.

Show the importance of the features

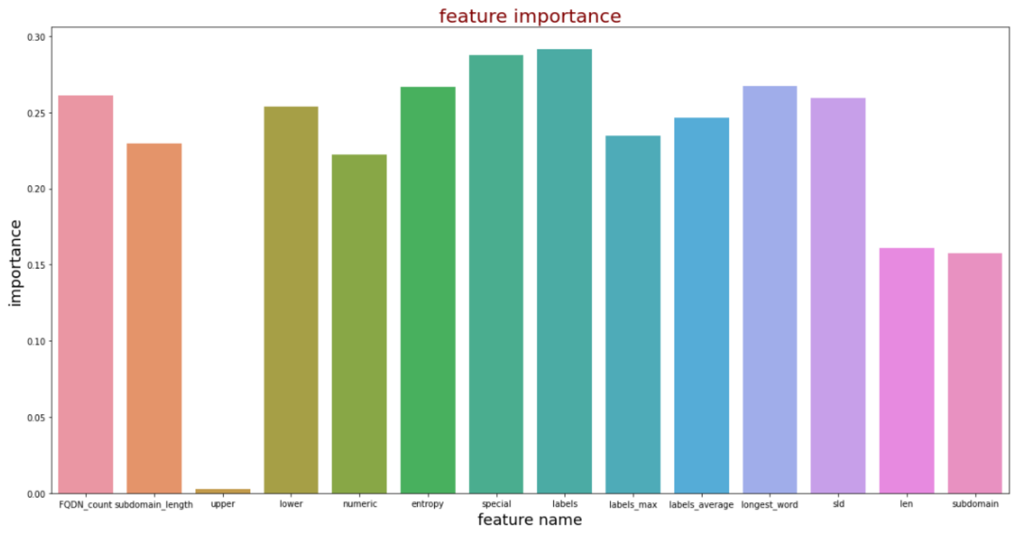


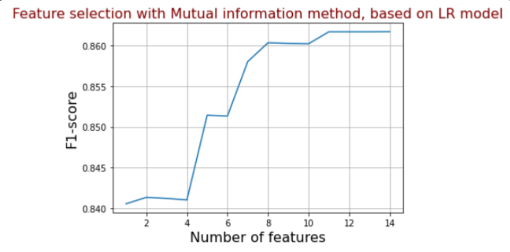
Figure 12: the importance of each feature

As the figure shows, some of the features are more important than others. for example, the "labels" and "special" features are the most important, and the "upper" feature is not as important as the others.

Check the correlation

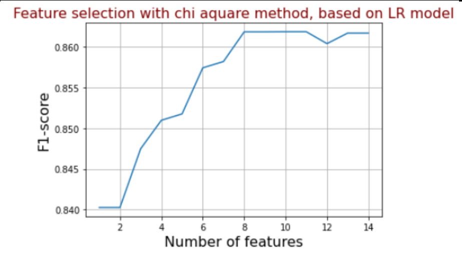
Figure (5) in the data analysis part shows to what extent the features are positively or negatively correlated to each other. Some features are highly correlated which means that I can select one and drop the other. One the other hand some features are highly correlated with the target which means that I must keep them since they affect the classification decision.

Train Logistic Regression with 3 different statistical methods.



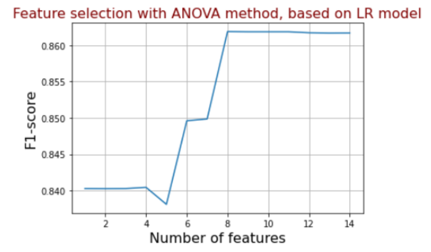
```
Maximum f1_score of LR model          : 0.8616863361796564
Best number of features for LR model : 14
```

Figure 13: feature selection based on logistic regression and Mutual information



```
Maximum f1_score of LR model          : 0.8618469408755476
Best number of features for LR model : 10
```

Figure 14: feature selection based on logistic regression with chi-square method



```
Maximum f1_score of LR model          : 0.8618722803716335
Best number of features for LR model : 8
```

Figure 15: feature selection based on logistic regression with ANOVA

I did not choose the features only based on the importance or even the correlation, I trained a simple classifier (Logistic Regression) first with mutual information method, second with chi-square, and finally with ANOVA test.
Based on the f1-score of the logistic regression classifier, figure (13) shows the best number of features with Mutual information is 14 features. Figure (14) shows that the best number of features with the chi-square method is 10 features. Figure (15) shows that the best number of features with the ANOVA test is 8 features.

Perform Dimensionality Reduction using PCA and tune the best number of components

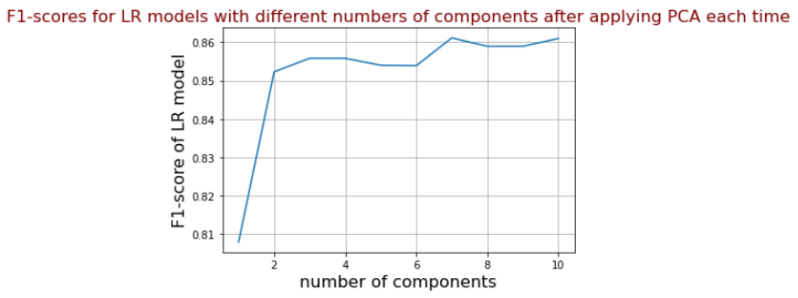


Figure 16: F1-score of the Logistic regression classifier after applying PCA on the data with different number of components

```
the best value of n_components, based on f1-score,for LR classifier
best F1-score is :( 0.8611 )%
best number of component is : 7
==========================================================
```

To study the effect of dimensionality reduction on the dataset, I tried PCA with different number of components.

As the figure shows, the best number of components is 7 components.

Compare the Three approaches and select the best features

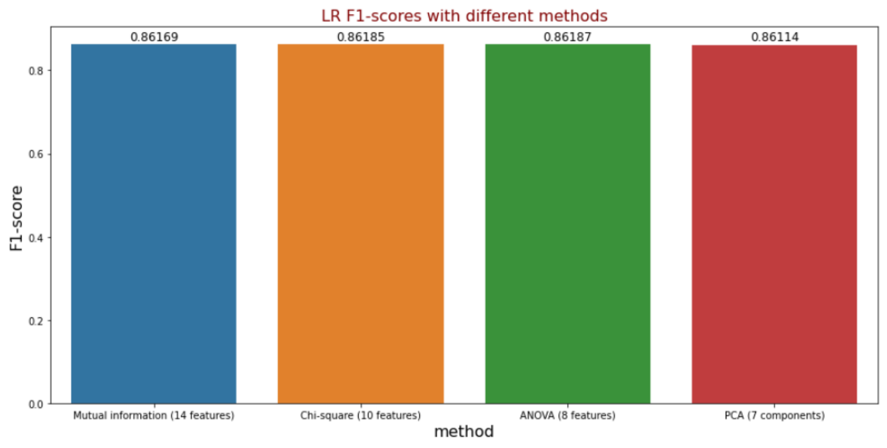


Figure 17: Comparison between the F1-score of Logistic regression classifier with different methods.

As the figure shows, the best number of features based on the F1-score of the logistic regression is 8 features. using ANOVA test.

```
ANOVA_selected_features_names
```

```
['FQDN_count',
 'subdomain_length',
 'lower',
 'numeric',
 'special',
 'labels',
 'longest_word',
 'subdomain']
```

Figure 18: the selected features.

4. Model Training

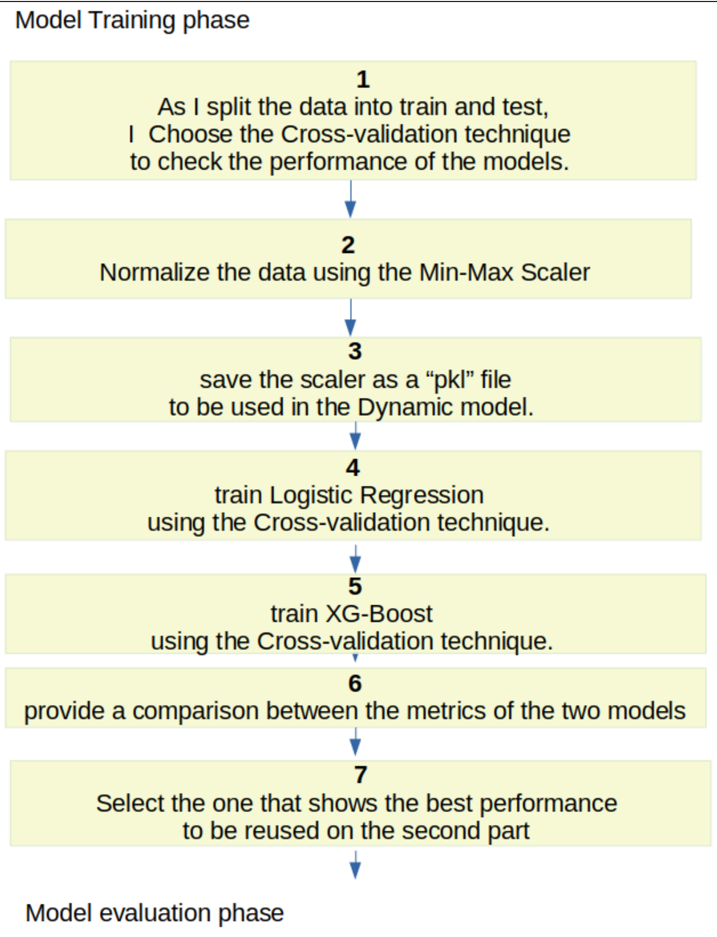

Model Training phase
1
As I split the data into train and test,
I Choose the Cross-validation technique
to check the performance of the models.
2
Normalize the data using the Min-Max Scaler
3
save the scaler as a “pkl” file
to be used in the Dynamic model.
4
train Logistic Regression
using the Cross-validation technique.
5
train XG-Boost
using the Cross-validation technique.
6
provide a comparison between the metrics of the two models
7
Select the one that shows the best performance
to be reused on the second part
Model evaluation phase

Figure 19: the model training phase.

I normalized the data to be on the same scale [0,1] using min-max scaler. I chose normalization not standardization because I wanted to scale the data to be in range [0,1] not to have a mean equal to zero and variance equal to one.

I saved the fitted scaler to be used in the next part (Dynamic Part) so I just transform the upcoming data, not to fit the scaler first and then transform since the upcoming data may have a new minimum value and new maximum value which give different scale than the model has already trained on, which lead the model to give low scores.

I have fitted both Logistic regression and XG-Boost classifier on the selected data after normalization. I chose Logistic regression and XG-Boost because I tried to study the effect of using a classifier like Logistic regression that gets affected by data normalization, and a classifier like XG-Boost that does not get affected by data normalization.

2. Normalize the data using the Min-Max scaler.

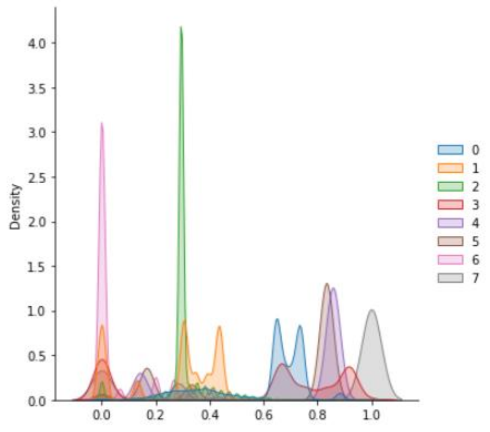
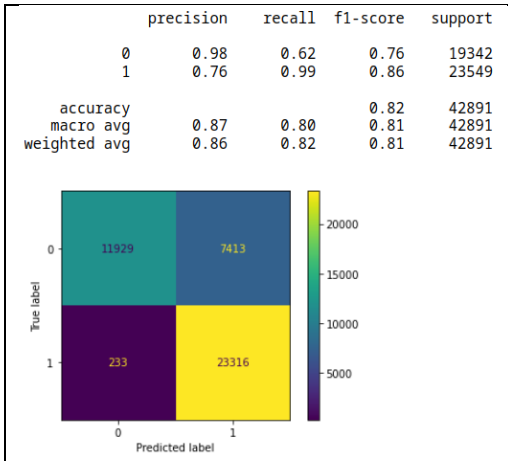


Figure 20: the distribution of the features after applying min-max scaler on the selected features.

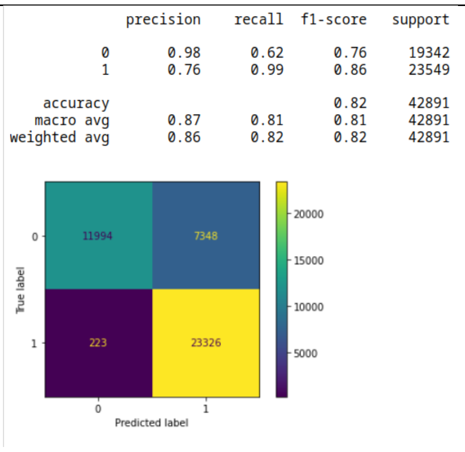
As the figure shows, the distribution of any feature after applying the scaler is in range [0,1].

4. train **Logistic Regression** using the Cross-validation technique.



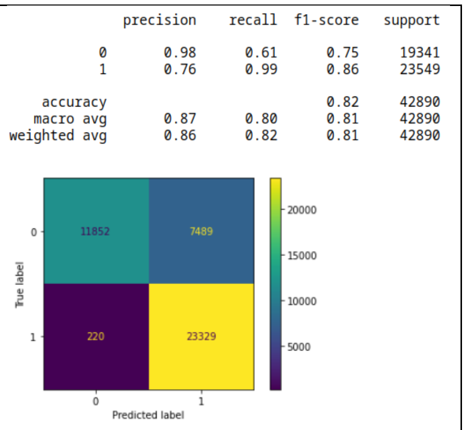
```
              precision    recall  f1-score   support

           0       0.98      0.62      0.76     19342
           1       0.76      0.99      0.86     23549

    accuracy                           0.82     42891
   macro avg       0.87      0.80      0.81     42891
weighted avg       0.86      0.82      0.81     42891
```

Figure 21: results of iteration 1



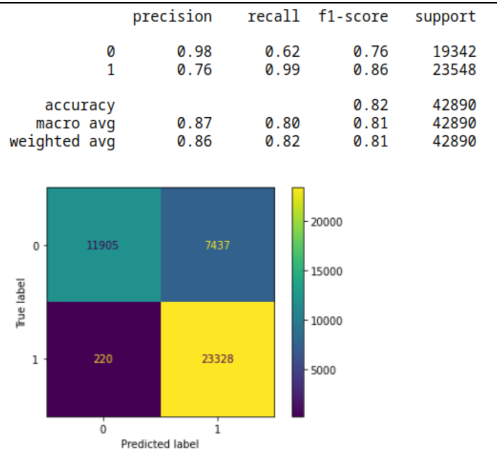
```
              precision    recall  f1-score   support

           0       0.98      0.62      0.76     19342
           1       0.76      0.99      0.86     23549

    accuracy                           0.82     42891
   macro avg       0.87      0.81      0.81     42891
weighted avg       0.86      0.82      0.82     42891
```

Figure 22: results of iteration 2



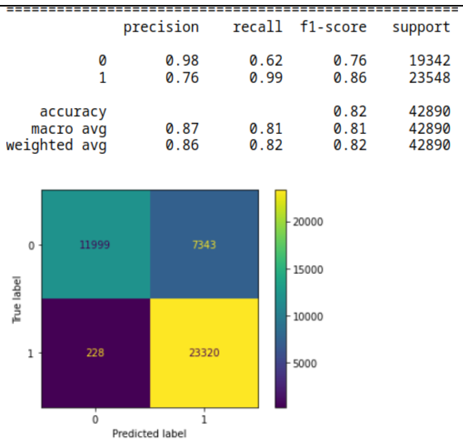
```
              precision    recall  f1-score   support

           0       0.98      0.61      0.75     19341
           1       0.76      0.99      0.86     23549

    accuracy                           0.82     42890
   macro avg       0.87      0.80      0.81     42890
weighted avg       0.86      0.82      0.81     42890
```

Figure 23: results of iteration 3



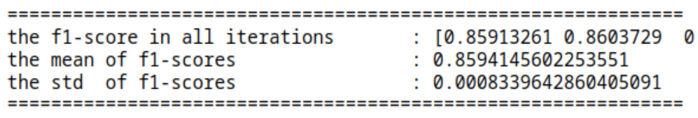
```
              precision    recall  f1-score   support

           0       0.98      0.62      0.76     19342
           1       0.76      0.99      0.86     23548

    accuracy                           0.82     42890
   macro avg       0.87      0.80      0.81     42890
weighted avg       0.86      0.82      0.81     42890
```

Figure 24: results of iteration 4



```
==============================================================
              precision    recall  f1-score   support

           0       0.98      0.62      0.76     19342
           1       0.76      0.99      0.86     23548

    accuracy                           0.82     42890
   macro avg       0.87      0.81      0.81     42890
weighted avg       0.86      0.82      0.82     42890
```

Figure 25: results of iteration 5

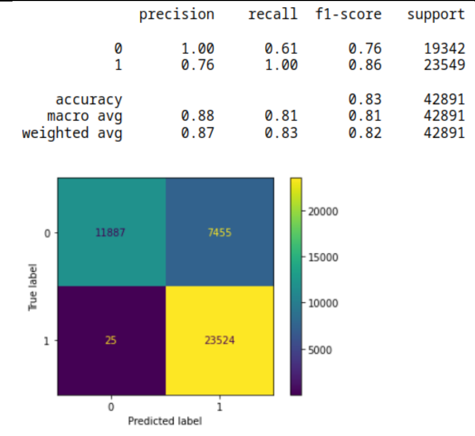


```
==============================================================
the f1-score in all iterations       : [0.85913261 0.8603729  0.85820443 0.85902086 0.860342  ]
the mean of f1-scores                : 0.8594145602253551
the std  of f1-scores                : 0.0008339642860405091
==============================================================
```

First I trained Logistic regression classifier using a 5-fold cross validation technique. I chose the F1-score of the classifier each time as an evaluation metric for the classifier Since the f1-score sums up between the precision and the recall. I did not choose accuracy because I care about the positive class which means that accuracy is not the best choice in this case because I may get a high accuracy, but the model fails to classify the positive instances.

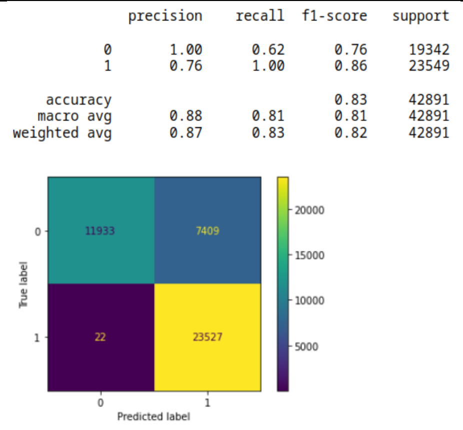
Figure (21 - 25) shows the results of each iteration using 5-fold cross validation technique.

5. train **XG-Boost** using the Cross-validation technique.



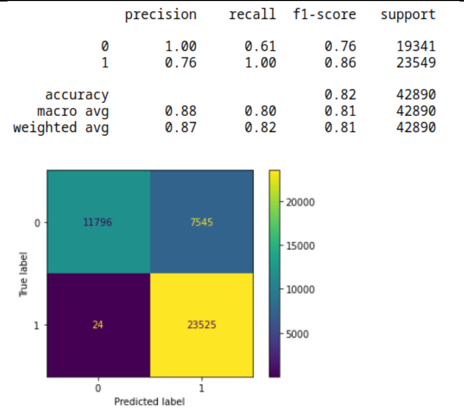
```
              precision    recall  f1-score   support

           0       1.00      0.61      0.76     19342
           1       0.76      1.00      0.86     23549

    accuracy                           0.83     42891
   macro avg       0.88      0.81      0.81     42891
weighted avg       0.87      0.83      0.82     42891
```

Figure 26: results of iteration 1



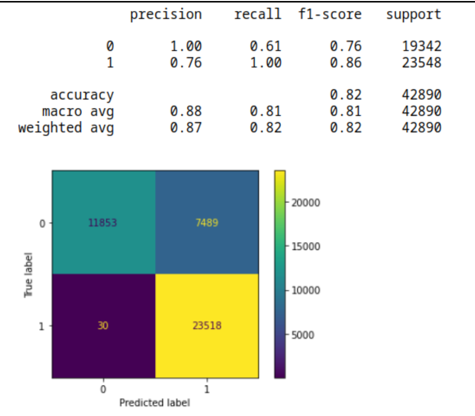
```
              precision    recall  f1-score   support

           0       1.00      0.62      0.76     19342
           1       0.76      1.00      0.86     23549

    accuracy                           0.83     42891
   macro avg       0.88      0.81      0.81     42891
weighted avg       0.87      0.83      0.82     42891
```

Figure 27: results of iteration 2



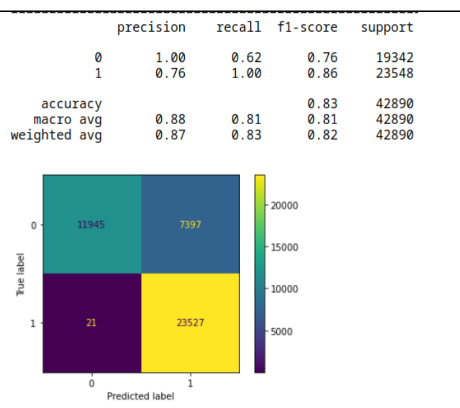
```
              precision    recall  f1-score   support

           0       1.00      0.61      0.76     19341
           1       0.76      1.00      0.86     23549

    accuracy                           0.82     42890
   macro avg       0.88      0.80      0.81     42890
weighted avg       0.87      0.82      0.81     42890
```

Figure 28: results of iteration 3



```
              precision    recall  f1-score   support

           0       1.00      0.61      0.76     19342
           1       0.76      1.00      0.86     23548

    accuracy                           0.82     42890
   macro avg       0.88      0.81      0.81     42890
weighted avg       0.87      0.82      0.82     42890
```

Figure 29: results of iteration 4



```
              precision    recall  f1-score   support

           0       1.00      0.62      0.76     19342
           1       0.76      1.00      0.86     23548

    accuracy                           0.83     42890
   macro avg       0.88      0.81      0.81     42890
weighted avg       0.87      0.83      0.82     42890
```

Figure 30: results of iteration 5

```
==============================================================
the f1-score in all iterations       : [0.86282277 0.86361384 0.86142185 0.86217579 0.86381994]
the mean of f1-scores                : 0.8627708376518856
the std  of f1-scores                : 0.0008930652522300761
==============================================================
```

As comparison between figure (21-25) and figure (26-30) the mean of f1-score of XG-Boost is greater than the mean Logistic regression.

The confusion matrix shows that XG-Boost false negative (which predicts positive instances as negative instances) numbers are less than Logistic Regression Since XG-Boost is in the range of 20 to 30, and Logistic regression is in the range of 200 to 300.

6. provide a comparison between the metrics of the two models.

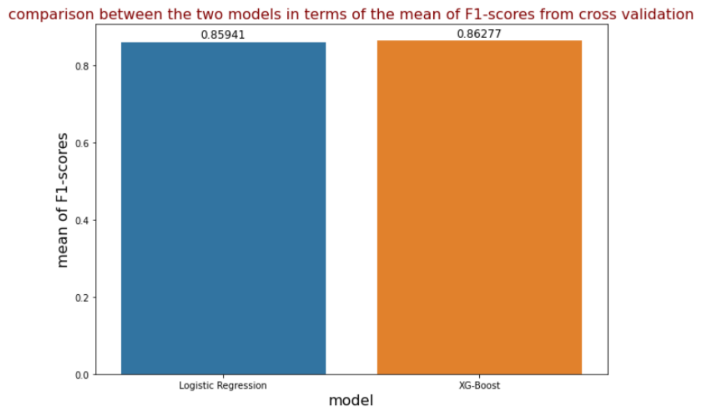
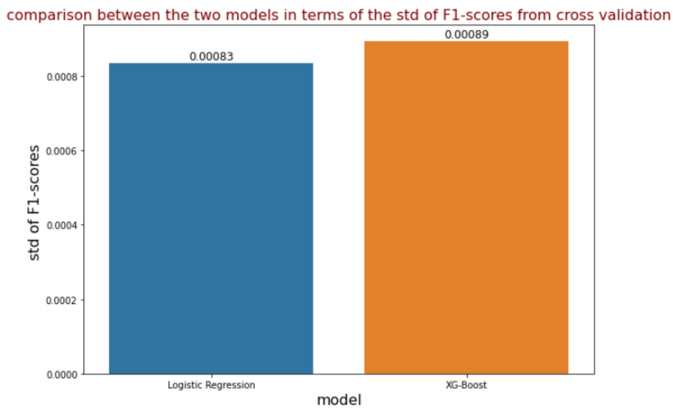
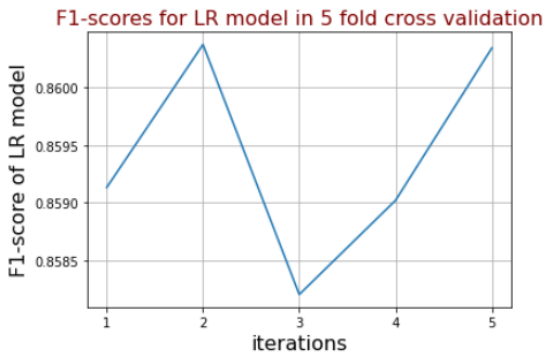
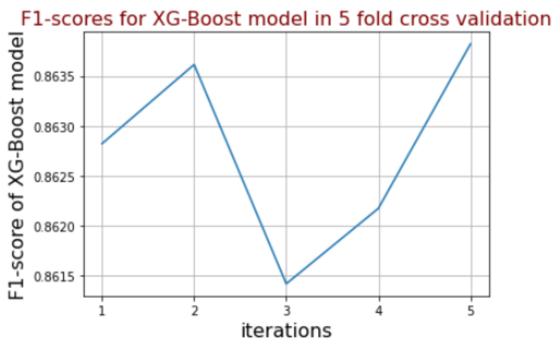
 Figure 31: comparison between Logistic regression and XG-Boost in terms of the mean of f1-scores over 5-fold cross validation	 Figure 32: comparison between Logistic regression and XG-Boost in terms of the standard deviation of f1-scores over 5-fold cross validation

Figure 31 shows that XG-Boost has mean of the values of the f1-score is greater than logistic regression model.
Figure 32 shows that XG-Boost has a standard deviation of the values of the f1-score greater than logistic regression model.
So, in terms of high value in each iteration the XG-Boost is better, but in terms of the value do not have a large variance to avoid overfitting the logistic regression is better.

7. Select the one that shows the best performance to be reused on the second part.

 Figure 33: F1-scores for Logistic regression model in 5-fold cross validation	 Figure 34: F1-scores for XG-Boost model in 5-fold cross validation

After checking the confusion matrix of each iteration in the cross validation, I chose the XG-Boost to be used in the next part (Dynamic model).
since the false negative values are less than the logistic regression. and the f1-score that sums up both the precision and the recall matrices is higher than the logistic regression.

5. Model evaluation

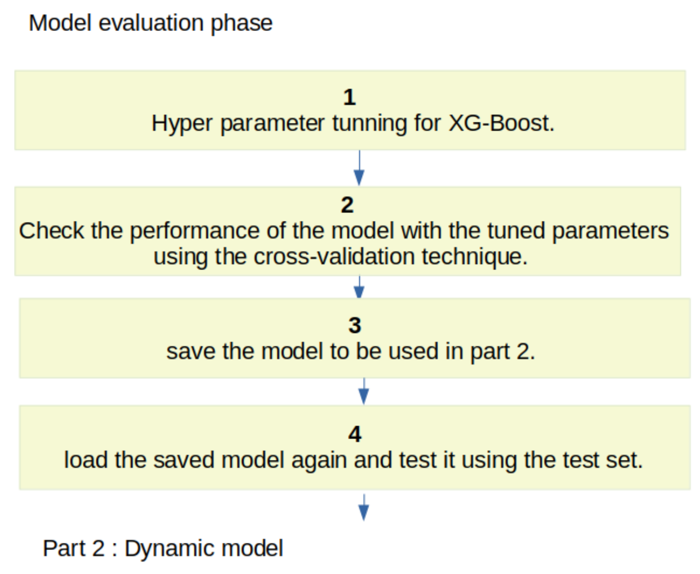

Model evaluation phase
1
Hyper parameter tunning for XG-Boost.
2
Check the performance of the model with the tuned parameters using the cross-validation technique.
3
save the model to be used in part 2.
4
load the saved model again and test it using the test set.
Part 2 : Dynamic model

Figure 35: the model evaluation phase

I tuned the parameters of the chosen classifier using a grid search technique. Then I built a classifier with the best parameters and checked its performance using a 5-fold cross-validation.

I just wanted to check its performance and check its confusion matrix each iteration. after evaluation the results of the cross-validation, I built a new model with the same best parameters from the grid search, I preferred to build a new classifier rather than the one from the cross-validation so it can be fitted in the entire "X_train" because in cross-validation will exclude one-fold for validation step.
I saved the model, then I loaded it again to test it using the test dataset, in this point specifically I used the accuracy as an evaluation metric, so it gives me a real representation of how the model performs with new data that it has never seen before.

1. Hyper parameter tuning for XG-Boost.

```
Best parameters from gridsearch: {'learning_rate': 0.1, 'max_depth': 10, 'n_estimators': 500}
CV score=0.863
```

Figure 36: the best parameters for XG-Boost model using Grid search with cross-validation technique

2. Check the performance of the model with the tuned parameters using the cross-validation technique.

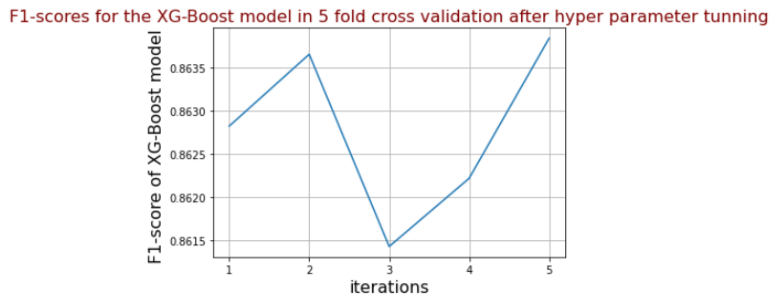
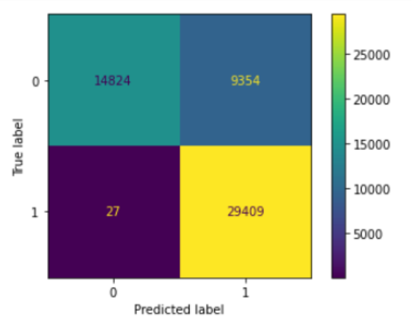


Figure 37: the value of F1-score in each iteration in cross-validation technique after tuning the parameters of the model



```
accuracy_score_value
0.8250270451747678
```

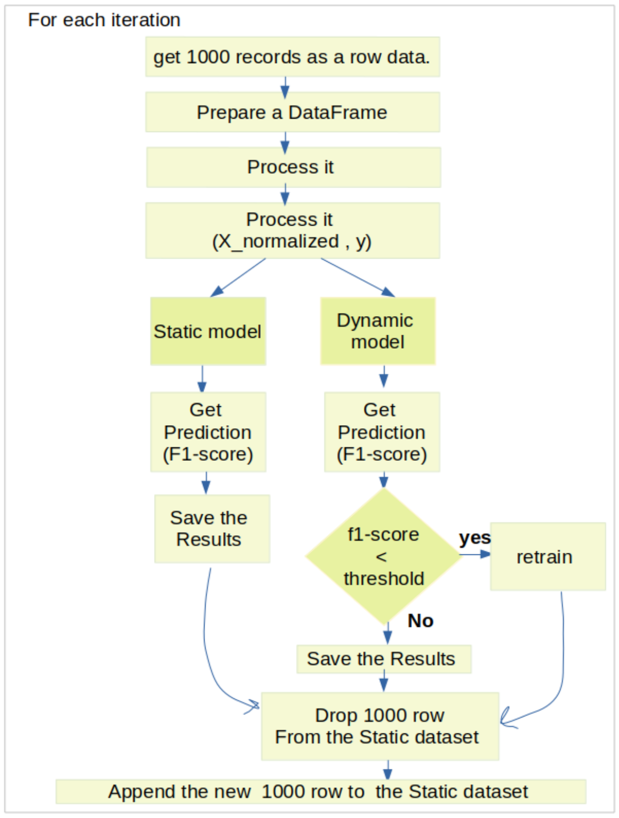
Figure 38: the confusion matrix of the XG-Boost on the test set

The XG-Boost give 82% accuracy on the test dataset.

Part II (Dynamic Model)

setup process

Install Dockers, create the images, install Kafka and the dependencies, run the producer code with Kafka_dataset, load the static model, run the code of the consumer, skip the first row to avoid errors while normalizing the data, and finally run the code that manages the experiment.



For each iteration
get 1000 records as a row data.
Prepare a DataFrame
Process it
Process it (X_normalized , y)
Static model
Dynamic model
Get Prediction (F1-score)
Get Prediction (F1-score)
Save the Results
f1-score < threshold
yes
retrain
No
Save the Results
Drop 1000 row From the Static dataset
Append the new 1000 row to the Static dataset

Figure 39: the core functionality for Dynamic model part

The purpose of this task is to simulate a real-life scenario of a constant data stream. to do that:

I have built a function to get a batch of 1000 rows from the “Kafka” dataset. The data comes in a row format, so I prepared a function to split the string and build a DataFrame with the same columns’ names as the static dataset. Then I built a function to drop null values, deal with the "longest_word" and "sld" columns and drop the "timestamp" column so the data will be passed to the models has the same data preprocessing as the data that the model has been trained on. Then I built a function to create x and y from the processed frame, then normalize x and return the “normalized_X” and “y”. In the normalization step, I used the saved scaler from part 1 so, I do not have to refit the scaler.

Once I got the normalized x and y, I prepared a loop in which I check both the static and the dynamic model and retrain the dynamic model if needed.

I decided to loop 249 times since the size of the dataset is larger than this number, but I was limited by the computing power of my local machine. Before choosing this number, I was assigning a large number so the last iteration in the loop never ends waiting the data to come in. That is the reason behind choosing this number.

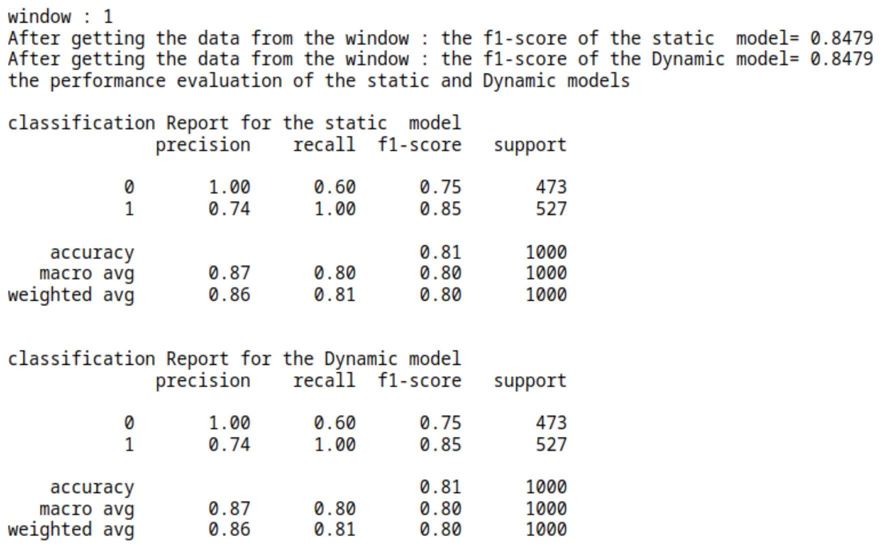
In each iteration, pass the data to test both the static and the dynamic model. I chose the f1-score to evaluate the models. since it sums up between the precision and the recall.

I decided to retrain the dynamic model again if the f1-score is not larger than 0.85.

I decided to update the static dataset by dropping the first 1000 rows each time and appending the new 1000 rows and decided to do that after the retraining phase to make sure that the model has never seen this data in the training phase.

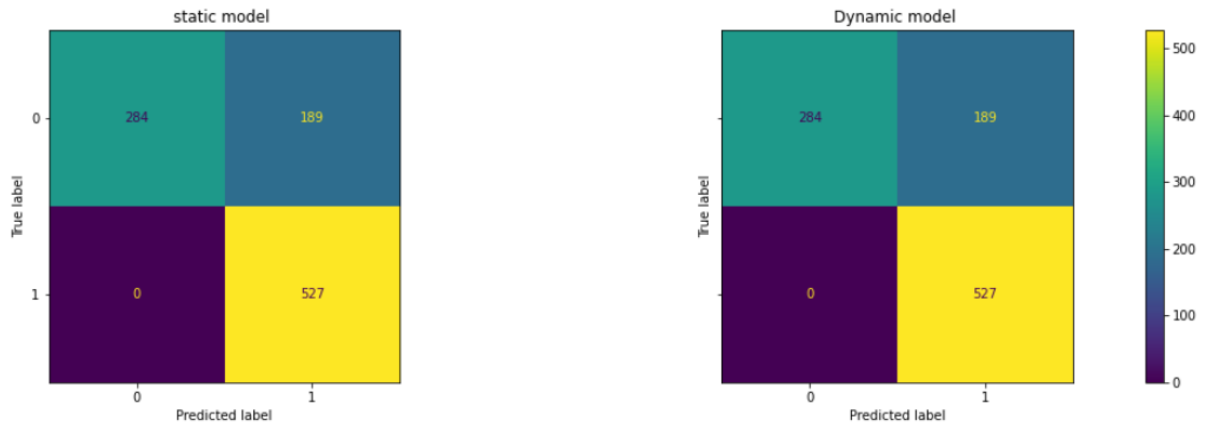
Evaluate the performance of each model on each window.

For each iteration I have printed the classification report and the confusion matrix for both the static model and the Dynamic model.



```
window : 1
After getting the data from the window : the f1-score of the static  model= 0.8479
After getting the data from the window : the f1-score of the Dynamic model= 0.8479
the performance evaluation of the static and Dynamic models

classification Report for the static  model
              precision    recall  f1-score   support

           0       1.00      0.60      0.75       473
           1       0.74      1.00      0.85       527

    accuracy                           0.81      1000
   macro avg       0.87      0.80      0.80      1000
weighted avg       0.86      0.81      0.80      1000

classification Report for the Dynamic model
              precision    recall  f1-score   support

           0       1.00      0.60      0.75       473
           1       0.74      1.00      0.85       527

    accuracy                           0.81      1000
   macro avg       0.87      0.80      0.80      1000
weighted avg       0.86      0.81      0.80      1000
```

Figure 40: the classification reports of the static and dynamic model at window (1)



```
 The f1-score value of the Dynamic model under the threshold , then retrain it
After Retrain with the (static_dataset + window data) : the f1-score of the Dynamic model = 0.8479
==========================================================================================
```

Figure 41: the confusion matrices of the static and dynamic model at window (1)

As expected in the first window both the static and Dynamic have the same results. but later they got different results.

make a conclusion of whether the dynamic implementation is better or not, compared to the traditional static model.

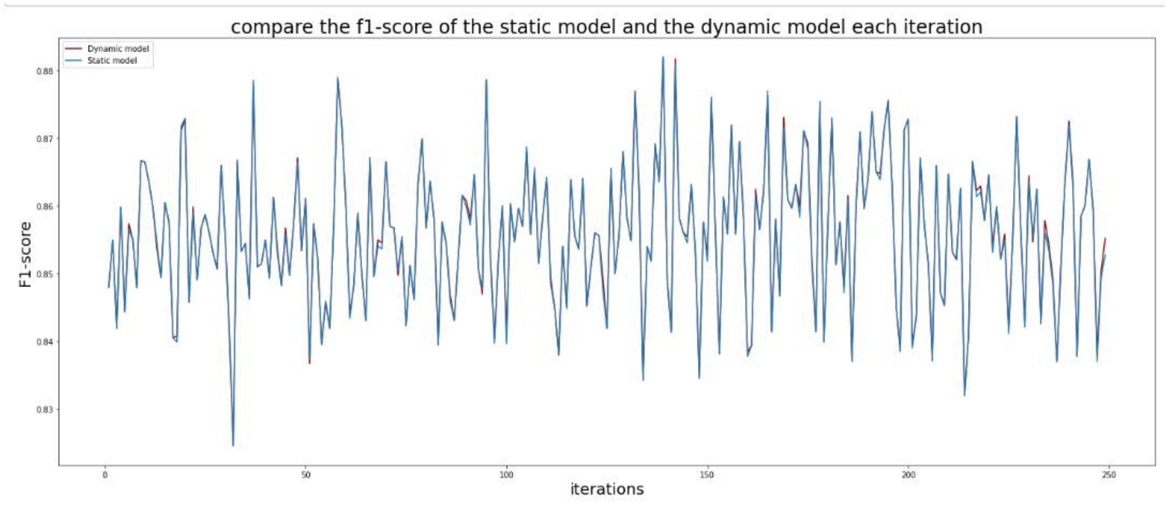
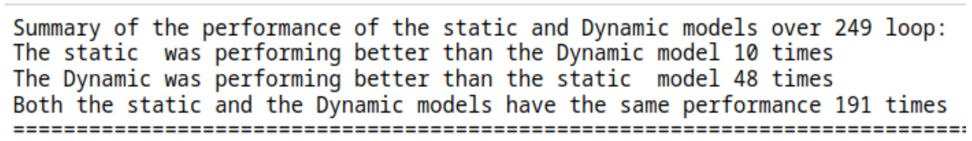


Figure 42: compare the results of the static and the dynamic model in each iteration



```
Summary of the performance of the static and Dynamic models over 249 loop:
The static  was performing better than the Dynamic model 10 times
The Dynamic was performing better than the static  model 48 times
Both the static and the Dynamic models have the same performance 191 times
===========================================================================
```

Figure 43: summary of the results

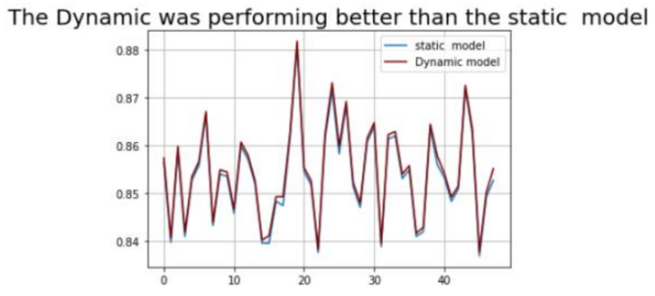


Figure 44:the iterations in which the Dynamic model was performing better than the static model

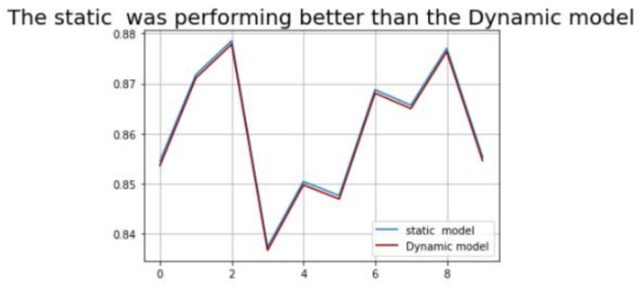


Figure 45:the iterations in which the static model was performing better than the dynamic model

Figure (44) shows that the Dynamic model was performing better than the static model 48 times.
Figure (45) shows that the static model was performing better than the dynamic model just 10 times.
The two models have had the same results 191 times.
As a conclusion the Dynamic model in total performs better than the static model.

References

[1] https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html#sphx-glr-auto-examples-preprocessing-plot-all-scaling-py

[2] https://www.geeksforgeeks.org/normalization-vs-standardization/

[3] https://hossainlab.github.io/dataviz/notebooks/SB01-Distribution%20Plots.html

[4] https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

[5] https://www.unb.ca/cic/datasets/dns-exf-2021.html

[6] https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/