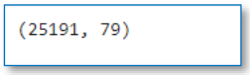
# ELG7186[EG] AI For Cyber Security 20219

## Dina Abdelhady

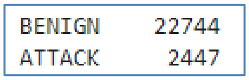
Kafka Assignment-Task 1

## Dataset:

I worked on “cicids_static_data.csv” which contains 25191 rows and 79 features 22744 over 25191 rows are labeled as “BENIGN” and only 2447 rows are labeled as “ATTACK”.



```
(25191, 79)
```



```
BENIGN     22744
ATTACK      2447
```

## Data preprocessing:

1. I use label encoding to convert the “Label” from string to integer type. 0 represents the “BENIGN” and 1 represents “ATTACK”
2. Then, clean the data from ant nan and infinity values.
3. Split the data into 80% train and 20% test

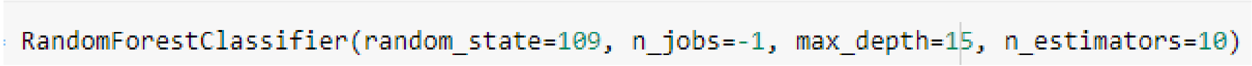
## Algorithms description:

- ### Random Forest:

Random Forest is bagging ensemble method

1. In Random Forest n number of random records are taken from the data set having k number of records. It is obtained from a bootstrap sample of the original data.
2. Random Forests consist of sets of tree-based models. Individual decision trees are constructed for each sample and uses some form of random selection of variables during tree growth.
3. Each decision tree will generate an output.
4. Final output is considered based on *Majority Voting* for Classification.

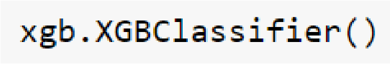
My parameters were



```
RandomForestClassifier(random_state=109, n_jobs=-1, max_depth=15, n_estimators=10)
```

I used n_jobs=-1 to run my code parallel on all processors to decrease the running time.
I used max_depth=15 the default is none that mean the model will train until all leaves are Prue.
Also, I put estimators=10 that mean my model will have 10 of decision trees will predict and vote The result. 10 models it is enough and run-in suitable time. Finally, I determine specific random state which random the bootstrapping. And the other parameters were on the default. -sckitlearn

- ### XGBClassifier



```
xgb.XGBClassifier()
```

It is *sequential ensemble* learning method to builder on new weak learners.
I used the default parameters Such as:
n_estimators=100 so, it takes longer time than the random forest. And learning_rate=*0.1*.

## Algorithm evaluation

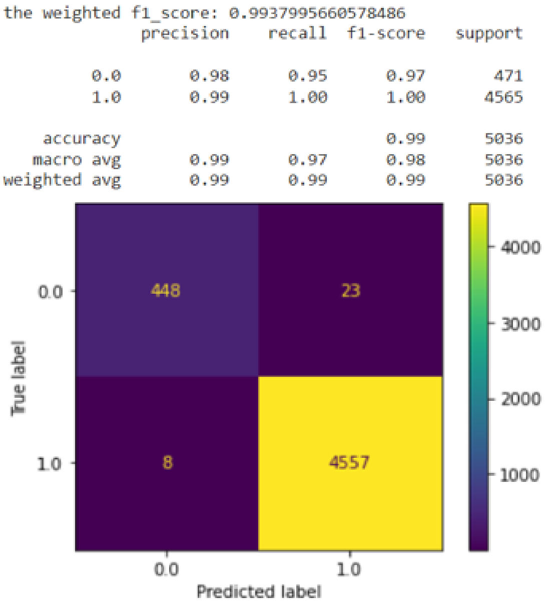
Regarding to figure 1: Random Forest:
The model predicted 23 of class 0 as class 1.
and predicted 8 of class 1 as class 0.

- ### XGBClassifier

Regarding to figure 2:
I used the default parameters Such as:
n_estimators=100 so, it takes longer time than the random forest. And learning_rate=*0.1*.

The model predicted 9 of class 0 as class 1.
And predicted 6 of class 1 as class 0.

The xgboost get higher f1_score but I choose the Random Forest as a static model.
And compare it's performance by dynamic random forest model and dynamic xgboost model

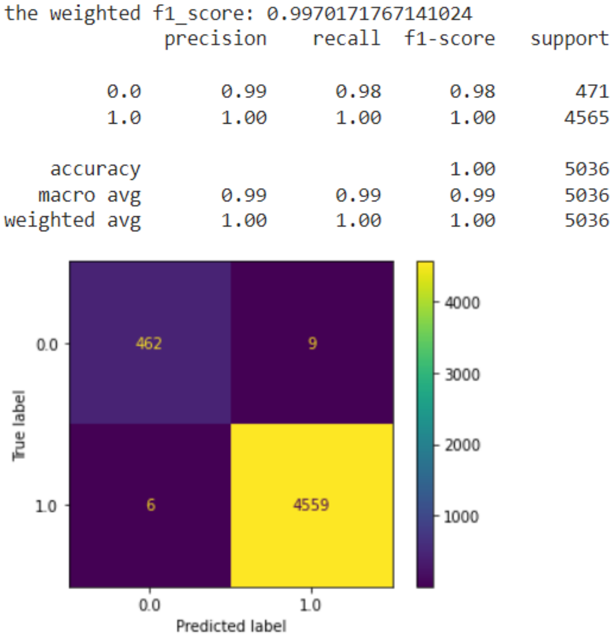


```
the weighted f1_score: 0.9937995660578486
              precision    recall  f1-score   support

         0.0       0.98      0.95      0.97       471
         1.0       0.99      1.00      1.00      4565

    accuracy                           0.99      5036
   macro avg       0.99      0.97      0.98      5036
weighted avg       0.99      0.99      0.99      5036
```

*Figure 1: Random Forest evaluation*

## Dynamic :

- After building 2 models, I stared to get 5000 packets from Kafka. Then, I tested the 3 models (static RF, dynamic RF and dynamic RF). To evaluation I used f1_score and the classification report. Then I started to tak the fixed window equall the size of the data by drop the first 5000 from the original data then append the new packets. The train the daynamic models on the new data.

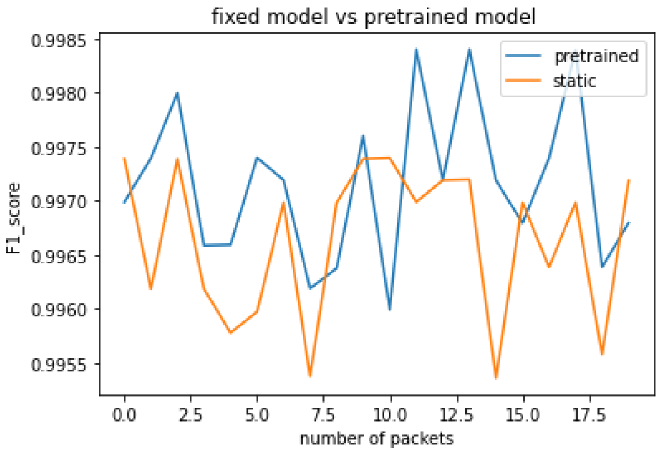


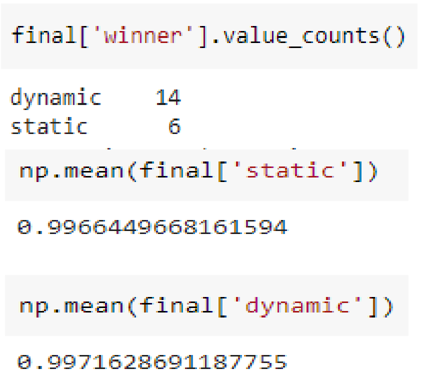
```
the weighted f1_score: 0.9970171767141024
              precision    recall  f1-score   support

         0.0       0.99      0.98      0.98       471
         1.0       1.00      1.00      1.00      4565

    accuracy                           1.00      5036
   macro avg       0.99      0.99      0.99      5036
weighted avg       1.00      1.00      1.00      5036
```

*Figure 2:XGboost Evaluation*

## Results discussion

### XGBOOST Dynamic comparing Random foreststatic model





```
final['winner'].value_counts()

dynamic     14
static       6
```

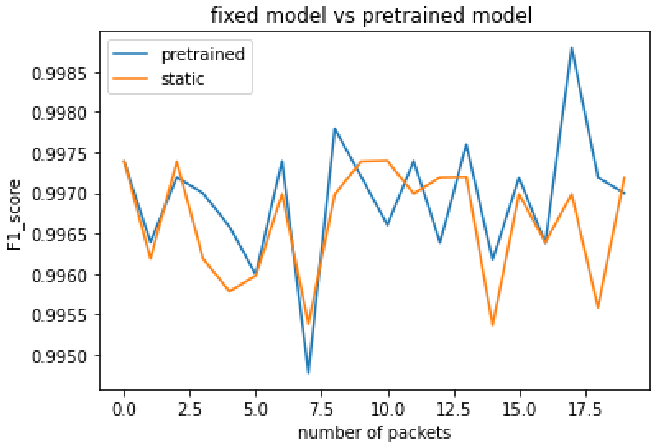
```
np.mean(final['static'])

0.9966449668161594
```

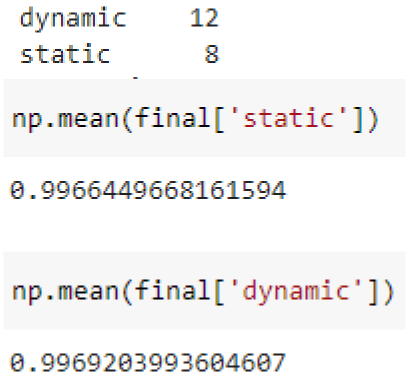
```
np.mean(final['dynamic'])

0.9971628691187755
```

The xgboost got the higher f1_score in 14 different iterations.
The mean of f1_score for xgboost is greater than the mean of the static model.

### RF comparing to static

The rf got the higher f1_score in 12 different iterations .





```
dynamic     12
static       8
```

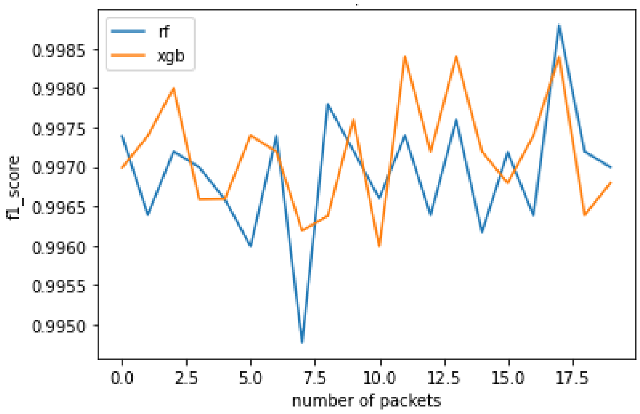
```
np.mean(final['static'])

0.9966449668161594
```

```
np.mean(final['dynamic'])

0.9969203993604607
```

The dynamic Random Forest model got the higher f1_score in 14 different iterations.
the mean of f1_score for namic Random Forest model is greater than the mean of the static model.

## Compare rf vs xgboost

In figure3, I plot the f1_score for both of Randam forest and the xgboost. The 2 models have aproxmatelty the same performance. The xgboost get high f1_score in different 12 itteration.



## Advantages&limitations

1. It took a long time to train the model for each iteration.
2. It the number of attacks' classes were small or there are classes not founded then I trained my model on this data that make the f1-score for the next iteration be low.
3. I prefer more apply trigger model instead of the fixed window "forgetting" which is pretrain If there are change in the data

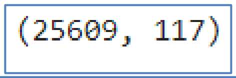
## knowledge learned

how to build ml algorithm on streaming data. And how make my model adaptive.

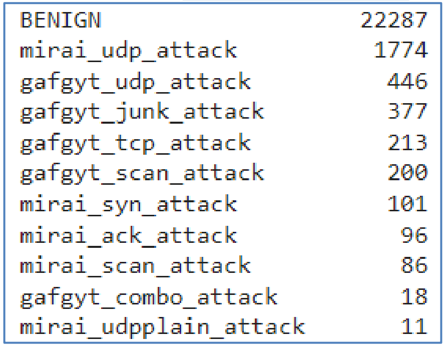
# Kafka Assignment-Task2

## Dataset:

I worked on "cicids_static_data.csv" which contains 25609rows and 117 features labeled as shownin figure 6.



```
(25609, 117)
```



```
BENIGN                  22287
mirai_udp_attack         1774
gafgyt_udp_attack         446
gafgyt_junk_attack        377
gafgyt_tcp_attack         213
gafgyt_scan_attack        200
mirai_syn_attack          101
mirai_ack_attack           96
mirai_scan_attack          86
gafgyt_combo_attack        18
mirai_udpplain_attack      11
```

*Figure 3: labeled*

## Data preprocessing:

1. I use label encoding to convert the "source" from string to integer type.
2. Split the data into 80% train and 20% test

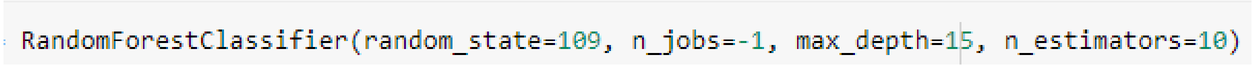
## Algorithms description:

- ### Random Forest:

Random Forest is bagging ensemble method

5. In Random Forest n number of random records are taken from the data set having k number of records. It is obtained from a bootstrap sample of the original data.
6. Random Forests consist of sets of tree-based models. Individual decision trees are constructed for each sample and uses some form of random selection of variables during tree growth.
7. Each decision tree will generate an output.
8. Final output is considered based on *Majority Voting* for Classification.

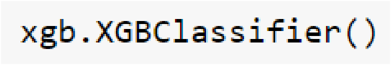
My parameters were



```
RandomForestClassifier(random_state=109, n_jobs=-1, max_depth=15, n_estimators=10)
```

I used n_jobs=-1 to run my code parallel on all processors to decrease the running time.
I used max_depth=15 the default is none that mean the model will train until all leaves are Prue. Also, I put estimators=10 that mean my model will have 10 of decision trees will predict and vote The result. 10 models it is enough and run-in suitable time. Finally, I determine specific random state which random the bootstrapping. And the other parameters were on the default. -sckitlearn

- **XGBClassifier**



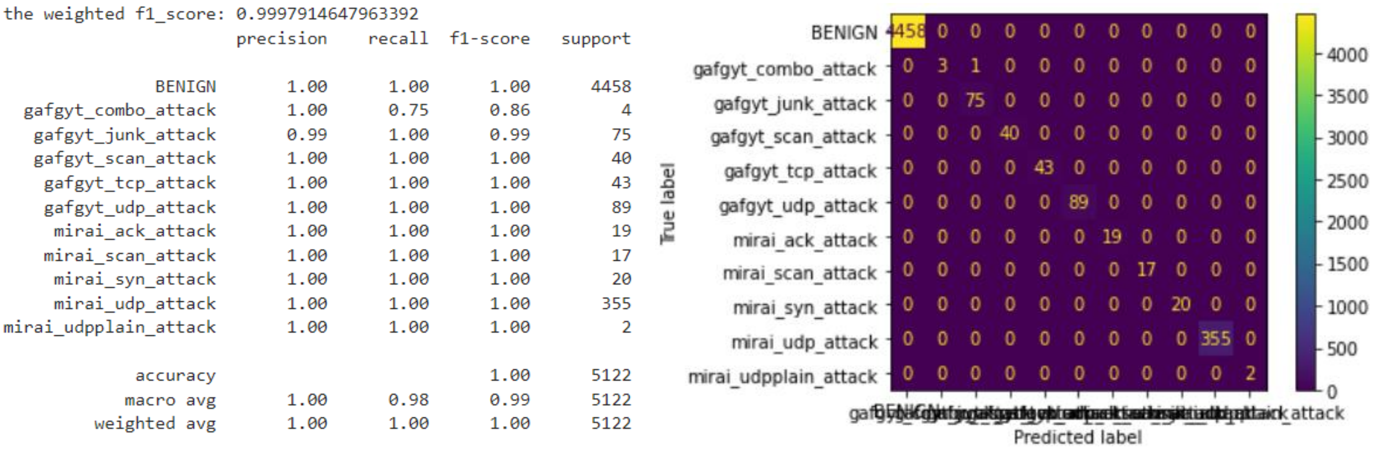
```
xgb.XGBClassifier()
```

It is *sequential ensemble* learning method to builder on new weak learners.
I used the default parameters Such as:
n_estimators=100 so, it takes longer time than the random forest. And learning_rate=*0.1*.

## Algorithm evaluation

### Random Forest:

The model can predict all class correctly except a row from class "gafgyt_combo_attack". It predicted it as "gafgyt_junk_attack".
So, all class have 1 in precision, recall and f1_score except these 2 classes.



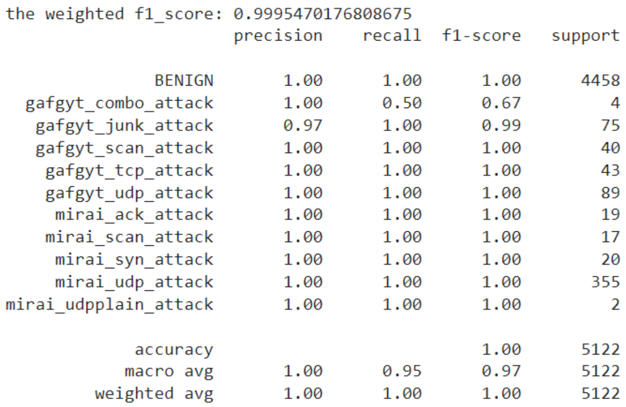
```
the weighted f1_score: 0.9997914647963392
                     precision    recall  f1-score   support

             BENIGN       1.00      1.00      1.00      4458
  gafgyt_combo_attack       1.00      0.75      0.86         4
   gafgyt_junk_attack       0.99      1.00      0.99        75
   gafgyt_scan_attack       1.00      1.00      1.00        40
    gafgyt_tcp_attack       1.00      1.00      1.00        43
    gafgyt_udp_attack       1.00      1.00      1.00        89
     mirai_ack_attack       1.00      1.00      1.00        19
    mirai_scan_attack       1.00      1.00      1.00        17
     mirai_syn_attack       1.00      1.00      1.00        20
     mirai_udp_attack       1.00      1.00      1.00       355
mirai_udpplain_attack       1.00      1.00      1.00         2

           accuracy                           1.00      5122
          macro avg       1.00      0.98      0.99      5122
       weighted avg       1.00      1.00      1.00      5122
```

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| BENIGN | 4458 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| gafgyt_combo_attack | 0 | 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| gafgyt_junk_attack | 0 | 0 | 75 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| gafgyt_scan_attack | 0 | 0 | 0 | 40 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| gafgyt_tcp_attack | 0 | 0 | 0 | 0 | 43 | 0 | 0 | 0 | 0 | 0 | 0 |
| gafgyt_udp_attack | 0 | 0 | 0 | 0 | 0 | 89 | 0 | 0 | 0 | 0 | 0 |
| mirai_ack_attack | 0 | 0 | 0 | 0 | 0 | 0 | 19 | 0 | 0 | 0 | 0 |
| mirai_scan_attack | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 17 | 0 | 0 | 0 |
| mirai_syn_attack | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 0 | 0 |
| mirai_udp_attack | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 355 | 0 |
| mirai_udpplain_attack | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |

*Figure 4: Random Forest evaluation*

### XGBClassifier

The model can predict all class correctly except two rows from class "gafgyt_combo_attack". It predicted it as "gafgyt_junk_attack".
So, all class have 1 in precision, recall and f1_score except these 2 classes. The recall of class "gafgyt_combo_attack"=0.5 that is because the model can predict only 2 of 4 rows.



```
the weighted f1_score: 0.9995470176808675
                      precision    recall  f1-score   support

              BENIGN       1.00      1.00      1.00      4458
  gafgyt_combo_attack       1.00      0.50      0.67         4
   gafgyt_junk_attack       0.97      1.00      0.99        75
   gafgyt_scan_attack       1.00      1.00      1.00        40
    gafgyt_tcp_attack       1.00      1.00      1.00        43
    gafgyt_udp_attack       1.00      1.00      1.00        89
     mirai_ack_attack       1.00      1.00      1.00        19
    mirai_scan_attack       1.00      1.00      1.00        17
     mirai_syn_attack       1.00      1.00      1.00        20
     mirai_udp_attack       1.00      1.00      1.00       355
mirai_udpplain_attack       1.00      1.00      1.00         2

             accuracy                           1.00      5122
            macro avg       1.00      0.95      0.97      5122
         weighted avg       1.00      1.00      1.00      5122
```

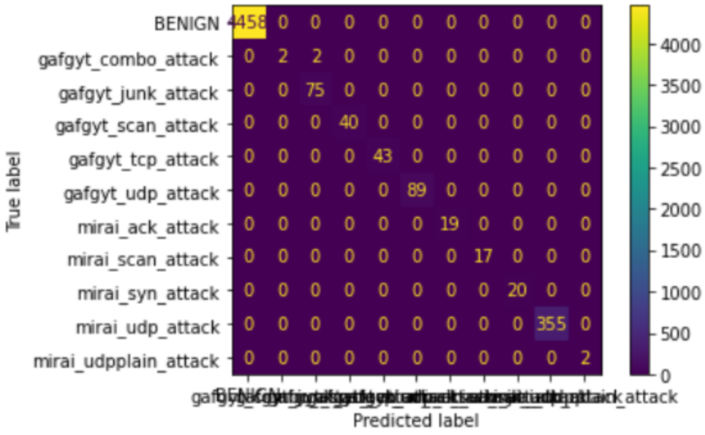
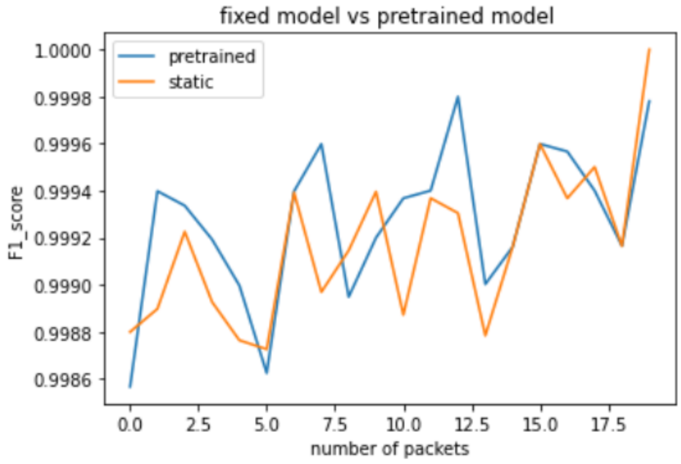


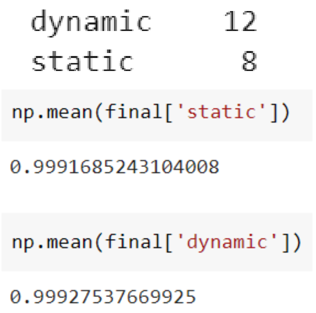
*Figure 5:XGboost Evaluation*

The Random Forest get higher f1_score so,Random Forest as a static model.
And compare it’s performance by dynamic random forest model and dynamic xgboost model.

## Dynamic :

### XGBOOST Dynamic vs Random Forest static model





```
dynamic    12
static      8
```

```
np.mean(final['static'])
```

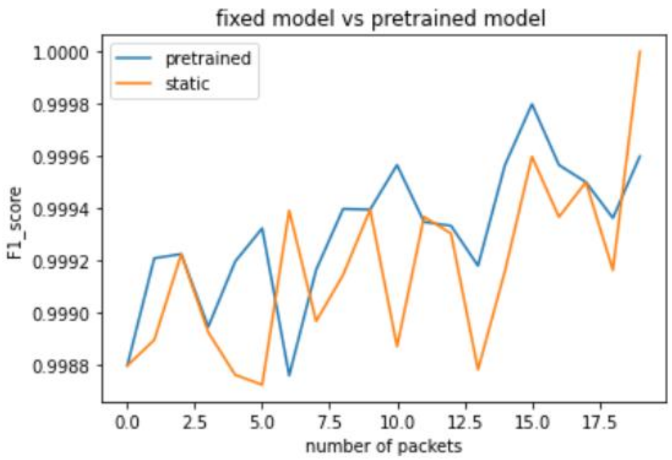
```
0.9991685243104008
```

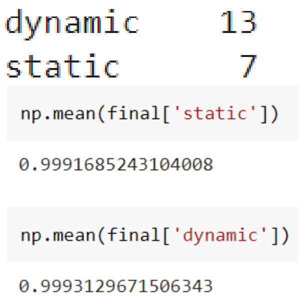
```
np.mean(final['dynamic'])
```

```
0.99927537669925
```

The xgboost got the higher f1_score in 12 different iterations.
The mean of f1_score for xgboost is greater than the mean of the static model.

## RF Dynamic vs Random Forest static model





```
dynamic     13
static       7
```

```
np.mean(final['static'])
```

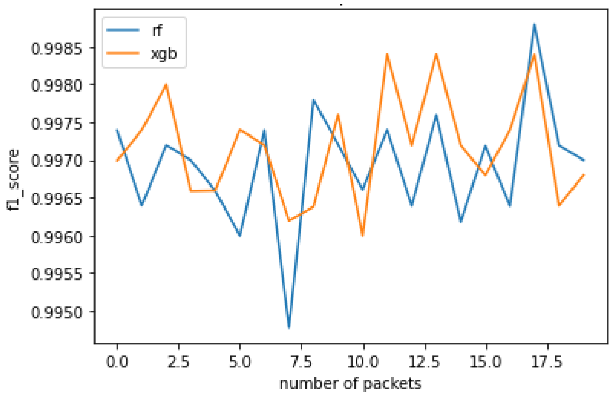
```
0.9991685243104008
```

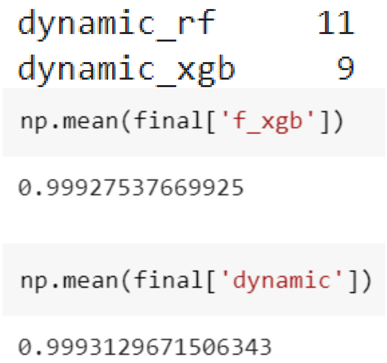
```
np.mean(final['dynamic'])
```

```
0.9993129671506343
```

The dynamic Random Forest model got the higher f1_score in 13 different iterations. the mean of f1_score for the  Random Forest model is greater than the mean of the static model.

## Random Forest  dynamic vs xgboost





```
dynamic_rf      11
dynamic_xgb      9
```

```
np.mean(final['f_xgb'])
```

```
0.99927537669925
```

```
np.mean(final['dynamic'])
```

```
0.9993129671506343
```

In figure3 , I plot the f1_score for both of rf and the xgboost. The 2 models have approximately the same performance. The RF get high f1_score in different 11 iterations

### Advantages&limitations

1. It took a long time to train the model for each iteration.
2. It the number of attacks' classes were small or there are classes not founded then I trained my model on this data that make the f1-score for the next iteration be low.
3. I think prefer to apply trigger model instead of the fixed window "forgetting".

### knowledge learned

how to build ml algorithm on streaming data. And how make my model adaptive.