# Assignment 3

## Description Continuous Evaluation with Kafka

### Part I (Static Model):

1. Data Analysis:

First, “Static_dataset.csv” was imported.

Second, the label "Target Attack" was discovered to determine whether or not it was balanced.

```
# validate if your dataset is imbalanced
datasetStatic['Target Attack'].value_counts()
```

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

In this Figure 1, "Target Attack" is balanced, and class 1 is more than class 0. Class 1 is important to achieve the goal of predicting attacks.

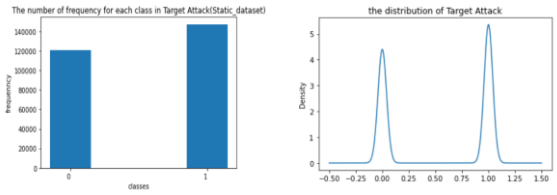
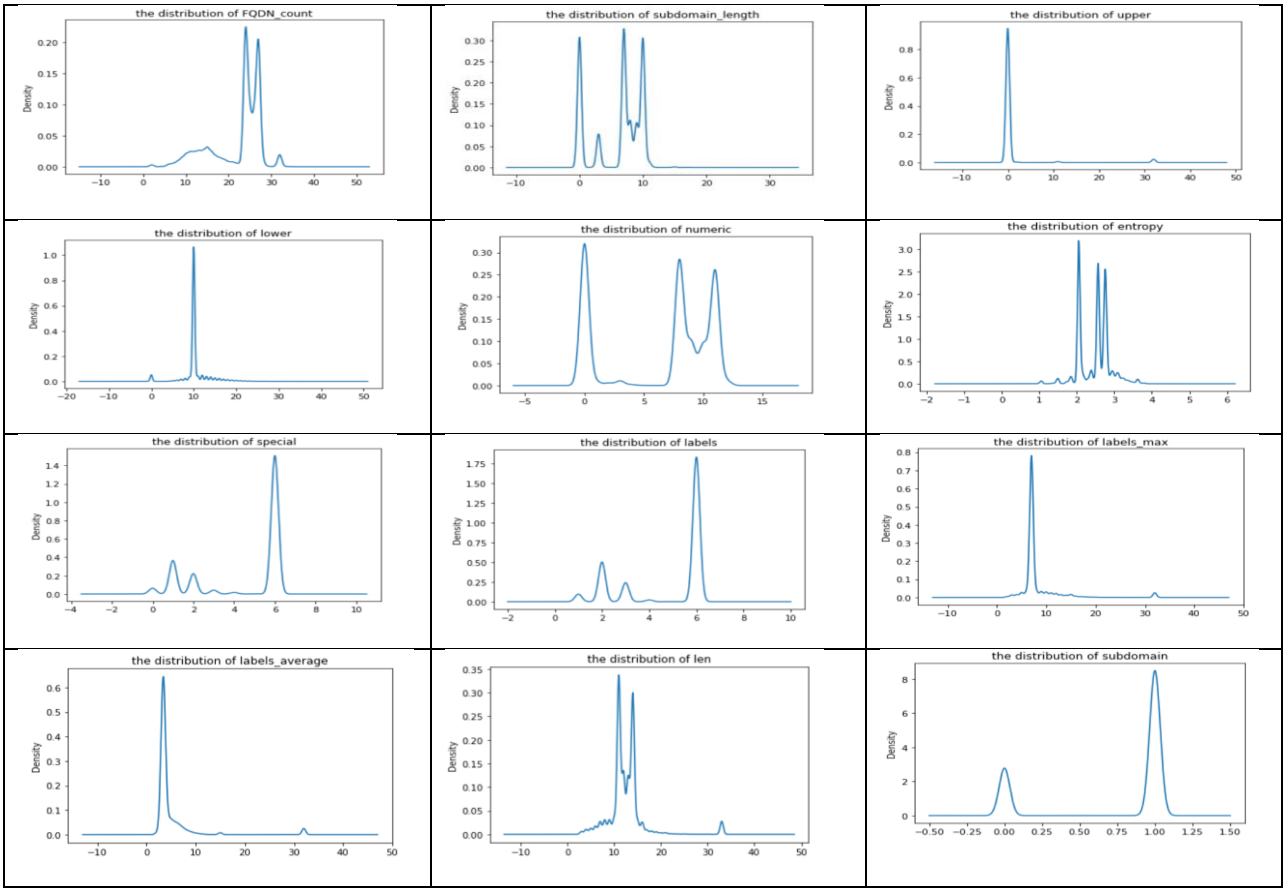



Figure 1: (**Target Attack**)

Third, each feature is categorical, so Kernel Density Estimate (**Kde**) was used to discover them.

```
# Plot the distribution of each feature and the target variable,
for f in datasetStaticOnlynumeric.columns:
    datasetStaticOnlynumeric[f].plot.kde()
    plt.title(f"the distribution of {f}")
    plt.show()
```




2. Feature engineering and data cleaning:

In this section, the "timestamp" column was dropped. "longest_word", "sld" columns contain string values that were replaced by the length of each word in each cell. Info () function was used to show information all features after cleaning.

```
datasetStaticWithoutTime['longest_word'] = datasetStaticWithoutTime['longest_word'].astype('str')
datasetStaticWithoutTime['longest_word'] =
        datasetStaticWithoutTime['longest_word'].apply(lambda x: x if any(c.isnumeric() for c in x) else len(x)).astype('int')
```

```
# Show information the features of "Static_dataset.csv" after cleaning
datasetStaticWithoutTime.info()
```

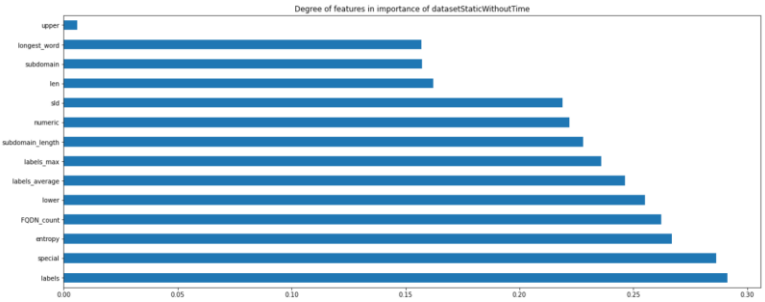
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 268074 entries, 0 to 268073
Data columns (total 15 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   FQDN_count        268074 non-null  int64
 1   subdomain_length  268074 non-null  int64
 2   upper             268074 non-null  int64
 3   lower             268074 non-null  int64
 4   numeric           268074 non-null  int64
 5   entropy           268074 non-null  float64
 6   special           268074 non-null  int64
 7   labels            268074 non-null  int64
 8   labels_max        268074 non-null  int64
 9   labels_average    268074 non-null  float64
 10  longest_word      268074 non-null  int32
 11  sld               268074 non-null  int32
 12  len               268074 non-null  int64
 13  subdomain         268074 non-null  int64
 14  Target Attack     268074 non-null  int64
dtypes: float64(2), int32(2), int64(11)
memory usage: 28.6 MB
```

```
# Show the number of null values in data
datasetStaticWithoutTime.isnull().sum()
```

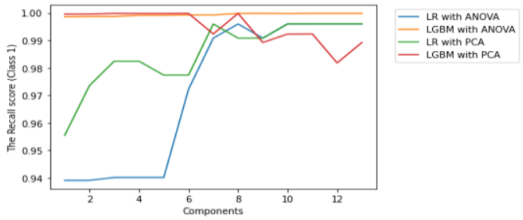
```
FQDN_count          0
subdomain_length    0
upper               0
lower               0
numeric             0
entropy             0
special             0
labels              0
labels_max          0
labels_average      0
longest_word        0
sld                 0
len                 0
subdomain 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

## 3. Feature Filtering/Selection:

First, datasetStaticWithoutTime was split into the features set “X” and label “y”.
Second, “mutual_info_classif” was used to determinate the important features.




Third, ANOVA and PCA were used to extract the important features by validation data.




| The Recall score (Class 1) of LR | | |
|---|---|---|
| ANOVA | 11 | 0.996002 |
| PCA | 11 | 0.996002 |

('ANOVA', 11) components was the best value based on the Recall score (Class 1) of the LogisticRegression classifier: 0.9960018974046215%

| The Recall score (Class 1) of LGBM | | |
|---|---|---|
| ANOVA | 9 | 0.999797 |
| PCA | 4 | 0.999797 |

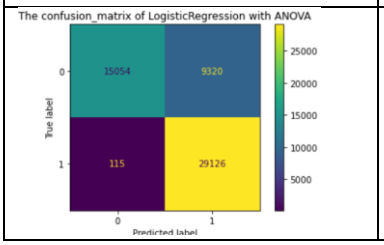
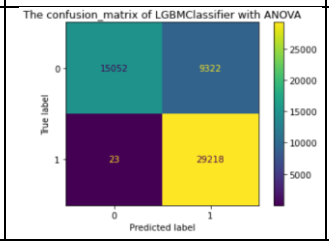
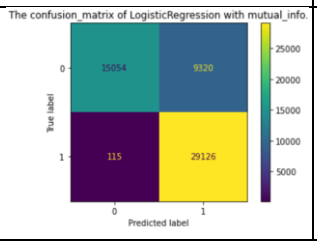
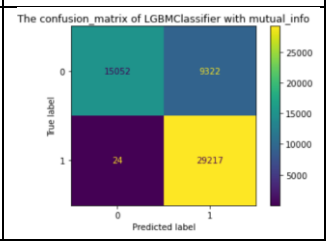
('ANOVA', 9) components was the best value based on the Recall score (Class 1) of the LGBMCla ssifier classifier: 0.9997967066476926%

## 4. Model Training & Model evaluation:

First, RobustScaler () was used to normalize the data before entering in the train.
Second, the feature extraction winner “ANOVA” was applied on normalized data.
Third, Grid Search was used on two models (“**logistic regression**” and “**LGBMClassifier**”) to get best parameters for them by the score Recall.

| Logistic Regression parameter distributions | LGBMClassifier parameter distributions |
|---|---|
| Best: 0.995876 using {'C': 100, 'penalty': 'l2', 'solver': 'newton-cg'} | Best: 0.999612 using {'min_child_samples': 100, 'num_leaves': 100, 'reg_alpha': 0.1} |
| 0.995876 (0.000636) with: {'C': 100, 'penalty': 'l2', 'solver': 'newton-cg'} | 0.999597 (0.000256) with: {'min_child_samples': 100, 'num_leaves': 100, 'reg_alpha': 0} |
| 0.995876 (0.000636) with: {'C': 100, 'penalty': 'l2', 'solver': 'lbfgs'} | 0.999612 (0.000254) with: {'min_child_samples': 100, 'num_leaves': 100, 'reg_alpha': 0.1} |
| 0.995876 (0.000636) with: {'C': 100, 'penalty': 'l2', 'solver': 'liblinear'} | 0.998515 (0.000465) with: {'min_child_samples': 100, 'num_leaves': 100, 'reg_alpha': 100} |
| 0.995872 (0.000631) with: {'C': 10, 'penalty': 'l2', 'solver': 'newton-cg'} | 0.999597 (0.000256) with: {'min_child_samples': 100, 'num_leaves': 150, 'reg_alpha': 0} |
| 0.995872 (0.000631) with: {'C': 10, 'penalty': 'l2', 'solver': 'lbfgs'} | 0.999612 (0.000254) with: {'min_child_samples': 100, 'num_leaves': 150, 'reg_alpha': 0.1} |
| 0.995872 (0.000631) with: {'C': 10, 'penalty': 'l2', 'solver': 'liblinear'} | 0.998515 (0.000465) with: {'min_child_samples': 100, 'num_leaves': 150, 'reg_alpha': 100} |
| 0.995872 (0.000631) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'newton-cg'} | 0.999597 (0.000256) with: {'min_child_samples': 100, 'num_leaves': 300, 'reg_alpha': 0} |
| 0.995872 (0.000631) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'lbfgs'} | 0.999612 (0.000254) with: {'min_child_samples': 100, 'num_leaves': 300, 'reg_alpha': 0.1} |
| 0.995872 (0.000631) with: {'C': 1.0, 'penalty': 'l2', 'solver': 'liblinear'} | 0.998515 (0.000465) with: {'min_child_samples': 100, 'num_leaves': 300, 'reg_alpha': 100} |
| 0.990806 (0.001082) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'newton-cg'} | 0.999593 (0.000252) with: {'min_child_samples': 150, 'num_leaves': 100, 'reg_alpha': 0} |
| 0.990806 (0.001082) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'lbfgs'} | 0.999593 (0.000267) with: {'min_child_samples': 150, 'num_leaves': 100, 'reg_alpha': 0.1} |
| 0.990806 (0.001082) with: {'C': 0.1, 'penalty': 'l2', 'solver': 'liblinear'} | 0.998519 (0.000461) with: {'min_child_samples': 150, 'num_leaves': 100, 'reg_alpha': 100} |
| 0.987398 (0.003390) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'newton-cg'} | 0.999593 (0.000252) with: {'min_child_samples': 150, 'num_leaves': 150, 'reg_alpha': 0} |
| 0.987398 (0.003390) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'lbfgs'} | 0.999593 (0.000267) with: {'min_child_samples': 150, 'num_leaves': 150, 'reg_alpha': 0.1} |
| 0.985200 (0.002128) with: {'C': 0.01, 'penalty': 'l2', 'solver': 'liblinear'} | 0.998519 (0.000461) with: {'min_child_samples': 150, 'num_leaves': 150, 'reg_alpha': 100} |
| | 0.999593 (0.000252) with: {'min_child_samples': 150, 'num_leaves': 300, 'reg_alpha': 0} |
| | 0.999593 (0.000267) with: {'min_child_samples': 150, 'num_leaves': 300, 'reg_alpha': 0.1} |
| | 0.998519 (0.000461) with: {'min_child_samples': 150, 'num_leaves': 300, 'reg_alpha': 100} |
| | 0.999582 (0.000243) with: {'min_child_samples': 300, 'num_leaves': 100, 'reg_alpha': 0} |
| | 0.999563 (0.000249) with: {'min_child_samples': 300, 'num_leaves': 100, 'reg_alpha': 0.1} |
| | 0.997342 (0.000536) with: {'min_child_samples': 300, 'num_leaves': 100, 'reg_alpha': 100} |
| | 0.999582 (0.000243) with: {'min_child_samples': 300, 'num_leaves': 150, 'reg_alpha': 0} |
| | 0.999563 (0.000249) with: {'min_child_samples': 300, 'num_leaves': 150, 'reg_alpha': 0.1} |
| | 0.997342 (0.000536) with: {'min_child_samples': 300, 'num_leaves': 150, 'reg_alpha': 100} |
| | 0.999582 (0.000243) with: {'min_child_samples': 300, 'num_leaves': 300, 'reg_alpha': 0} |
| | 0.999563 (0.000249) with: {'min_child_samples': 300, 'num_leaves': 300, 'reg_alpha': 0.1} |
| | 0.997342 (0.000536) with: {'min_child_samples': 300, 'num_leaves': 300, 'reg_alpha': 100} |

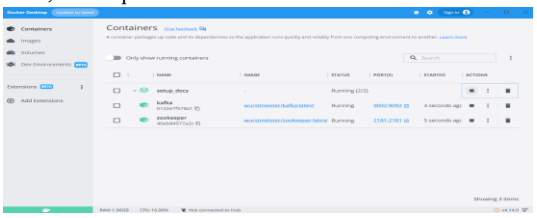
Fourth, two models were trained and evaluated using cross-validation using the score recall.

| **Logistic Regression**<br>With ANOVA (**11 components**)<br>- Mean score: **0.995872**<br>- A standard deviation **0.000129** | **LGBM Classifier**<br>With ANOVA ( **9 components**)<br>- Mean score: **0.999615**<br>- A standard deviation: **0.000068** | **Logistic Regression**<br>With mutual_info (**13 features**)<br>- Mean score: **0.990512**<br>- A standard deviation: **0.009279** | **LGBM Classifier**<br>With mutual_info (**13 features**)<br>-Mean score: **0.999604**<br>-A standard deviation: **0.000067** |
|---|---|---|---|
|  |  |  |  |

Finally, the LGBMClassifier Model with ANOVA is the winner because its mean score of recall (class 1) and standard deviation were the best, and the false positive of predicted class 0 was the least (23).

## Part II (Dynamic Model):

First, All requirement were installed and created like Dockers, Kafka, the dependencies and the images.



```
producer = KafkaProducer(bootstrap_servers=['localhost:9092'],
                         value_serializer=lambda x:
                         dumps(x).encode('utf-8'))
if producer.bootstrap_connected():
    print(f"Successfully connected to bootstrap server")
else:
    print("Couldn't connect to bootstrap server.")

TOPIC_NAME = "ml-raw-dns"
```

```
Successfully connected to bootstrap server
```

Second, the .csv file “Kafka_dataset.csv” was imported.

```
with open("Kafka_dataset.csv") as f:
    start_time = datetime.now()
    for i, line in tqdm(enumerate(f)):
        #print(i, line)
        produce_message(producer_instance=producer, topic=TOPIC_NAME, message=line)
    end_time = datetime.now()
    print(f"Batch took {end_time-start_time} time for ingesting data")

print("Ingestion Completed")
```

```
268065it [12:19, 362.52it/s]

Batch took 0:12:19.461757 time for ingesting data
Ingestion Completed
```

Third, the pipeline of champion static model “LGBMClassifier with ANOVA” was loaded. Static and dynamic models were created.

```
static_model = Dynamic_model= joblib.load('piplineModelLGBM_final.pkl')
Dynamic_model
```

```
Pipeline(steps=[('robustscaler', RobustScaler()),
                ('selectkbest', SelectKBest(k=9)),
                ('lgbmclassifier',
                 LGBMClassifier(min_child_samples=100, num_leaves=100,
                                random_state=42, reg_alpha=0.1))])
```

RobustScaler
RobustScaler()

SelectKBest
SelectKBest(k=9)

LGBMClassifier
LGBMClassifier(min_child_samples=100, num_leaves=100, random_state=42, reg_alpha=0.1)

Fourth, the consumer’s code was run and validated the data stream that was receiving.

```
consumer = KafkaConsumer(
    'ml-raw-dns',
    bootstrap_servers="localhost:9092",
    auto_offset_reset='earliest',
    enable_auto_commit=False
)
```

```
for m in consumer:
        x=m.value
        break
```

Fifth, 1,000 observations of data streaming information were appended in list and used as a window.

```
def getrecord_1000(itr):
    list_with_1000_record=[]
    i=0
    for m in consumer:

        if i <1000:
            list_with_1000_record.append(m.value)
            i=i+1
        else:

            break
    print(f"Window {itr}")
    return list_with_1000_record
```

Sixth, the dynamic model's performance was evaluated by the recall score. If it was less than the static model's recall score, the dynamic model was retrained.

```
for itr in range(267):
    r_dataset = getrecord_1000(itr+1)
    p_dataset = prepare_data(r_dataset)
    new_dataset =  data_cleaning(p_dataset)
    X =new_dataset.drop(labels = ["Target Attack"], axis=1)
    y=new_dataset["Target Attack"]
    #-ew_dataset = pd.concat([datasetStaticWithoutTime, new_dataset])
    print(new_dataset.shape)
    Dy_pred=Dynamic_model.predict(X)
    D_recall= recall_score(y, Dy_pred)
    D_acc= accuracy_score(y, Dy_pred)
    print(f"the Recall score (Class 1)  of Dynamic Model without retrain = {D_recall*100}%")
    print(f"the accuracy_score of Dynamic Model without retrain = {D_acc*100}%")
    new_dataset = pd.concat([datasetStaticWithoutTime, new_dataset])
    if D_recall < 0.999615 :
        print(" The model will be trained on the new data")
        Dynamic_model=retrain_model(new_dataset)
        Dy_pred=Dynamic_model.predict(X)
        D_recall=recall_score(y, Dy_pred)
        D_acc= accuracy_score(y, Dy_pred)
        print(f"the Recall score (Class 1)  of Dynamic Model after retrain = {D_recall*100}%")
        print(f"the accuracy_score of Dynamic Model after retrain = {D_acc*100}%")
```

```
Window 5
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.44649446494465%
the accuracy_score of Dynamic Model without retrain = 79.9%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.63099630996311%
the accuracy_score of Dynamic Model after retrain = 80.0%
the Recall score (Class 1)  of Static Model = 99.44649446494465%
the accuracy_score of Static Model = 79.9%
```

Seventh, the performance of each model was evaluated on each window.

```
Window 1
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.80620155038760%
the accuracy_score of Dynamic Model without retrain = 79.7%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.80620155038760%
the accuracy_score of Dynamic Model after retrain = 79.7%
the Recall score (Class 1)  of Static Model = 99.80620155038760%
the accuracy_score of Static Model = 79.7%
**********
Window 2
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 79.4%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 79.4%
**********
Window 3
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.82269503546l%
the accuracy_score of Dynamic Model without retrain = 82.3%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.82269503546l%
the accuracy_score of Dynamic Model after retrain = 82.3%
the Recall score (Class 1)  of Static Model = 99.82269503546l%
the accuracy_score of Static Model = 82.3%
**********
Window 4
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.81684981684981%
the accuracy_score of Dynamic Model without retrain = 81.10000000000001%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.81684981684981%
the accuracy_score of Dynamic Model after retrain = 81.10000000000001%
the Recall score (Class 1)  of Static Model = 99.81684981684981%
the accuracy_score of Static Model = 81.10000000000001%
**********
Window 5
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.44649446494465%
the accuracy_score of Dynamic Model without retrain = 79.9%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.63099630996311%
the accuracy_score of Dynamic Model after retrain = 80.0%
the Recall score (Class 1)  of Static Model = 99.44649446494465%
the accuracy_score of Static Model = 79.9%
**********
Window 6
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 81.6%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 81.6%
**********

Window 19
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 81.89999999999999%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 81.89999999999999%
**********
Window 20
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.82238010657194%
the accuracy_score of Dynamic Model without retrain = 80.9%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.82238010657194%
the accuracy_score of Dynamic Model after retrain = 80.9%
the Recall score (Class 1)  of Static Model = 99.82238010657194%
the accuracy_score of Static Model = 80.9%
**********
Window 21
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.81751824817519%
the accuracy_score of Dynamic Model without retrain = 82.6999999999999%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.81751824817519%
the accuracy_score of Dynamic Model after retrain = 82.6999999999999%
the Recall score (Class 1)  of Static Model = 99.81751824817519%
the accuracy_score of Static Model = 82.6999999999999%
**********
Window 22
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 81.10000000000001%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 81.0%
**********

Window 23
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.63099630996311%
the accuracy_score of Dynamic Model without retrain = 81.2%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.63099630996311%
the accuracy_score of Dynamic Model after retrain = 81.2%
the Recall score (Class 1)  of Static Model = 99.63099630996311%
the accuracy_score of Static Model = 81.2%
**********
Window 24
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 81.6%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 81.6%
**********

Window 249
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.81096408317580%
the accuracy_score of Dynamic Model without retrain = 80.1000000000001%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.81096408317580%
the accuracy_score of Dynamic Model after retrain = 80.1000000000001%
the Recall score (Class 1)  of Static Model = 99.81096408317580%
the accuracy_score of Static Model = 80.1000000000001%
**********
Window 250
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.6383363471971l%
the accuracy_score of Dynamic Model without retrain = 81.39999999999999%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.6383363471971l%
the accuracy_score of Dynamic Model after retrain = 81.39999999999999%
the Recall score (Class 1)  of Static Model = 99.6383363471971l%
the accuracy_score of Static Model = 81.39999999999999%
**********
Window 251
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.81785063752277%
the accuracy_score of Dynamic Model without retrain = 82.1999999999999%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.81785063752277%
the accuracy_score of Dynamic Model after retrain = 82.1999999999999%
the Recall score (Class 1)  of Static Model = 99.81785063752277%
the accuracy_score of Static Model = 82.1999999999999%
**********
Window 252
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 82.8%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 82.8%
**********
Window 253
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.81273408239l%
the accuracy_score of Dynamic Model without retrain = 80.7%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.81273408239l%
the accuracy_score of Dynamic Model after retrain = 80.7%
the Recall score (Class 1)  of Static Model = 99.81273408239l%
the accuracy_score of Static Model = 80.7%
**********

Window 7
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 80.4%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 80.4%
**********
Window 8
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.83193277310924%
the accuracy_score of Dynamic Model without retrain = 82.6999999999999%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.83193277310924%
the accuracy_score of Dynamic Model after retrain = 82.6999999999999%
the Recall score (Class 1)  of Static Model = 99.83193277310924%
the accuracy_score of Static Model = 82.6999999999999%
**********
Window 9
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 82.3%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 82.3%
**********
Window 10
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 80.4%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 80.4%
**********
Window 11
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.64093357271095%
the accuracy_score of Dynamic Model without retrain = 82.6%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.64093357271095%
the accuracy_score of Dynamic Model after retrain = 82.6%
the Recall score (Class 1)  of Static Model = 99.64093357271095%
the accuracy_score of Static Model = 82.6%
**********
Window 12
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.64601769911503%
the accuracy_score of Dynamic Model without retrain = 82.8999999999999%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.64601769911503%
the accuracy_score of Dynamic Model after retrain = 82.8999999999999%
the Recall score (Class 1)  of Static Model = 99.64601769911503%
the accuracy_score of Static Model = 82.8999999999999%
**********
Window 25
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 82.8%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 82.8%
**********
Window 26
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 79.9%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 79.9%
**********
Window 27
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.63833634719711%
the accuracy_score of Dynamic Model without retrain = 82.6%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.8191681735985l%
the accuracy_score of Dynamic Model after retrain = 82.6999999999999%
the Recall score (Class 1)  of Static Model = 99.63833634719711%
the accuracy_score of Static Model = 82.6%
**********
Window 28
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 81.39999999999999%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 81.39999999999999%
**********
Window 29
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.64726631393297%
the accuracy_score of Dynamic Model without retrain = 80.8%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.64726631393297%
the accuracy_score of Dynamic Model after retrain = 80.8%
the Recall score (Class 1)  of Static Model = 99.64726631393297%
the accuracy_score of Static Model = 80.8%
**********
Window 30
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.81751824817519%
the accuracy_score of Dynamic Model without retrain = 80.30000000000001%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.81751824817519%
the accuracy_score of Dynamic Model after retrain = 80.30000000000001%
the Recall score (Class 1)  of Static Model = 99.81751824817519%
the accuracy_score of Static Model = 80.30000000000001%
**********

Window 254
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 79.0%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 79.0%
**********
Window 255
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.82046678635548%
the accuracy_score of Dynamic Model without retrain = 82.5%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.82046678635548%
the accuracy_score of Dynamic Model after retrain = 82.5%
the Recall score (Class 1)  of Static Model = 99.82046678635548%
the accuracy_score of Static Model = 82.5%
**********
Window 256
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.8191681735985l%
the accuracy_score of Dynamic Model without retrain = 80.30000000000001%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 100.0%
the accuracy_score of Dynamic Model after retrain = 80.4%
the Recall score (Class 1)  of Static Model = 99.8191681735985l%
the accuracy_score of Static Model = 80.30000000000001%
**********
Window 257
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 81.10000000000001%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 81.2%
**********
Window 258
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 82.6%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 82.6%
**********
Window 259
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 83.5%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 83.5%
**********

Window 13
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.81583793738490%
the accuracy_score of Dynamic Model without retrain = 81.0%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.81583793738490%
the accuracy_score of Dynamic Model after retrain = 81.0%
the Recall score (Class 1)  of Static Model = 99.81583793738490%
the accuracy_score of Static Model = 81.0%
**********
Window 14
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 79.6000000000001%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 79.5%
**********
Window 15
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 83.39999999999999%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 83.39999999999999%
**********
Window 16
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 80.5%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 80.5%
**********
Window 17
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 79.7%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 79.7%
**********
Window 18
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.81617647058823%
the accuracy_score of Dynamic Model without retrain = 81.5%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.81617647058823%
the accuracy_score of Dynamic Model after retrain = 81.5%
the Recall score (Class 1)  of Static Model = 99.81617647058823%
the accuracy_score of Static Model = 81.5%
**********

Window 31
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 80.30000000000001%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 80.30000000000001%
**********
Window 32
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.80620155038760%
the accuracy_score of Dynamic Model without retrain = 81.3%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.80620155038760%
the accuracy_score of Dynamic Model after retrain = 81.3%
the Recall score (Class 1)  of Static Model = 99.80620155038760%
the accuracy_score of Static Model = 81.3%
**********
Window 33
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 82.1%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 82.1%
**********
Window 34
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 81.2%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 81.2%
**********
Window 35
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 82.1%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 82.1%
**********
Window 36
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 81.39999999999999%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 81.39999999999999%
**********
Window 37
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 82.39999999999999%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 82.39999999999999%
**********

Window 262
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.81308411214953%
the accuracy_score of Dynamic Model without retrain = 81.0%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.81308411214953%
the accuracy_score of Dynamic Model after retrain = 81.0%
the Recall score (Class 1)  of Static Model = 99.81308411214953%
the accuracy_score of Static Model = 81.0%
**********
Window 263
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 81.39999999999999%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 81.39999999999999%
**********
Window 264
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 78.8%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 78.8%
**********
Window 265
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 81.6%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 81.6%
**********
Window 266
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 100.0%
the accuracy_score of Dynamic Model without retrain = 81.8%
the Recall score (Class 1)  of Static Model = 100.0%
the accuracy_score of Static Model = 81.8%
**********
Window 267
(1000, 15)
the Recall score (Class 1)  of Dynamic Model without retrain = 99.61464354527938%
the accuracy_score of Dynamic Model without retrain = 81.8%
 The model will be trained on the new data
the Recall score (Class 1)  of Dynamic Model after retrain = 99.61464354527938%
the accuracy_score of Dynamic Model after retrain = 81.8%
the Recall score (Class 1)  of Static Model = 99.61464354527938%
the accuracy_score of Static Model = 81.8%
**********
```

Eighth, two lists were used to store the performance of both models, which were then plotted to show a comparison of both performances from both models.

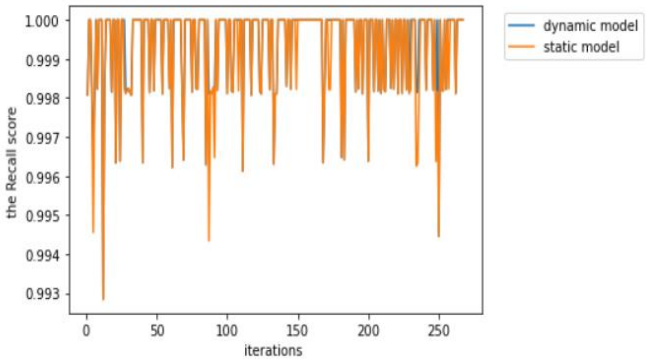



Figure (**The performance recall score of class 1 for both models on each window**)

This figure showed that the performance of the dynamic model was good with the prediction of class 1, but its accuracy was lower on each window because class 0 was smaller than class 1 in the training data. So the recall (class 1) was used to correctly predict attacks.

## A conclusion:

**In some windows, the recall score (class 1) of the dynamic model without retraining wasn't unsatisfactory, so it was retrained to get the best result. For example, in window 5, the accuracy was 79.9%; after retraining, it became 80%. The dynamic model (The recall score before and after retraining: 99.44649446494465% and 99.63099630996311%) and static model (the Recall score (Class 1) of Static Model = 99.44649446494465%, the accuracy score of Static Model = 79.9%). So it must be retrained.**