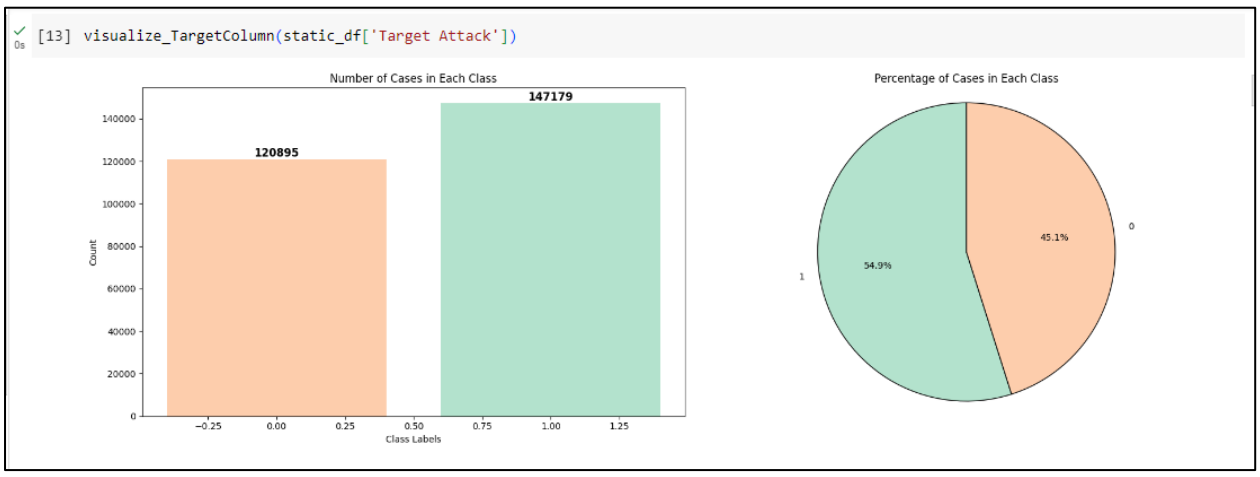
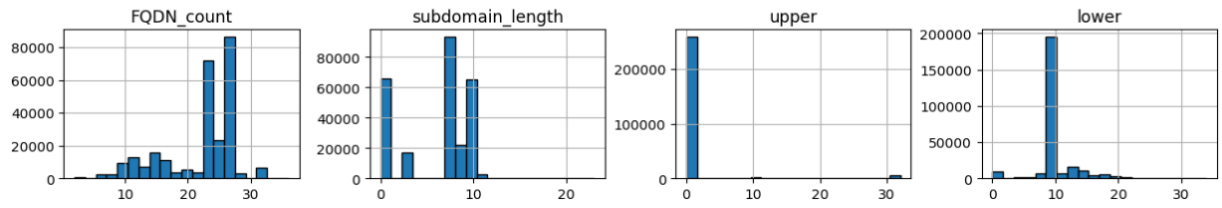
Kafka Assignment (3)

❖ Static Model

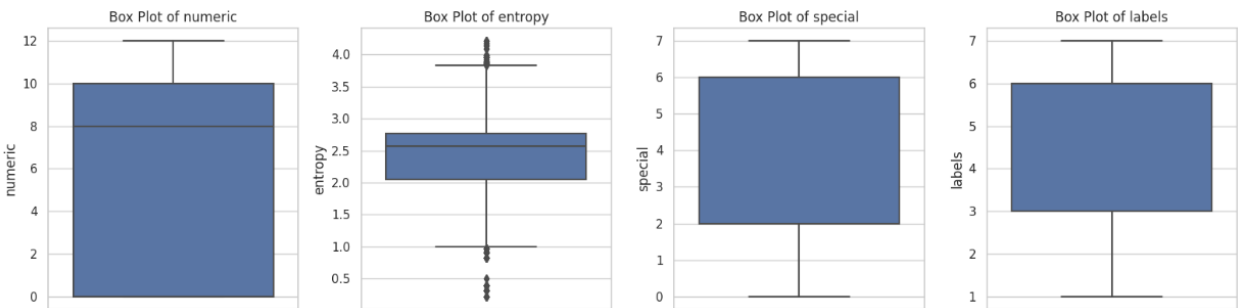
- **Imbalanced Data:** Imbalanced datasets can pose challenges for machine learning models, especially when the model is biased towards the majority class. In such cases, the model may have a tendency to predict the majority class more often, and this can lead to suboptimal performance, particularly for the minority class.



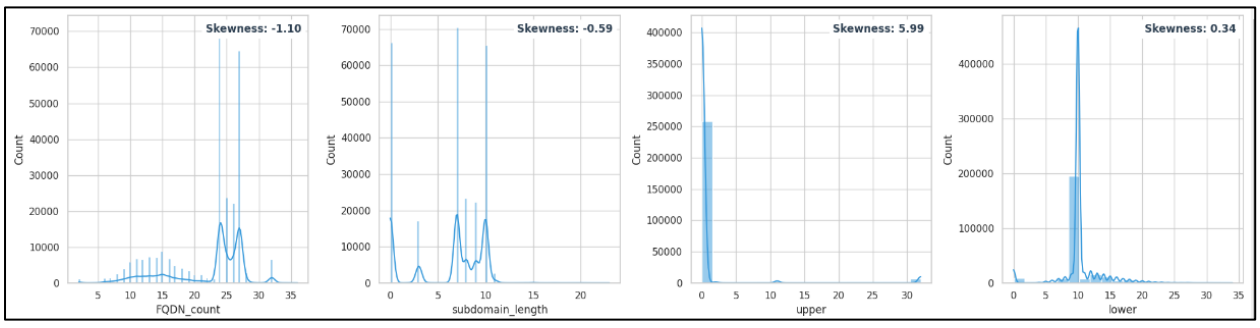
- **Histogram Analysis:** It shows the frequency distribution of values in different bins.



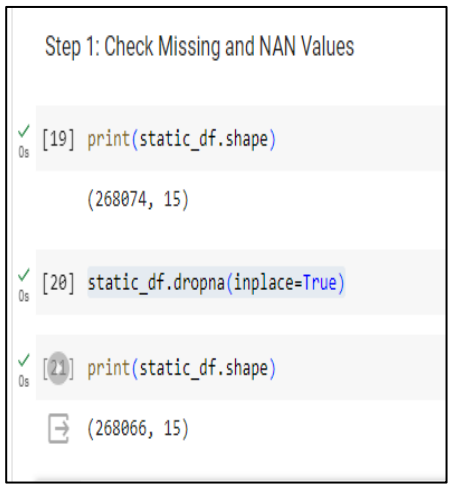
- **Box Plot Analysis:** This indicates that there are no data points considered outliers.

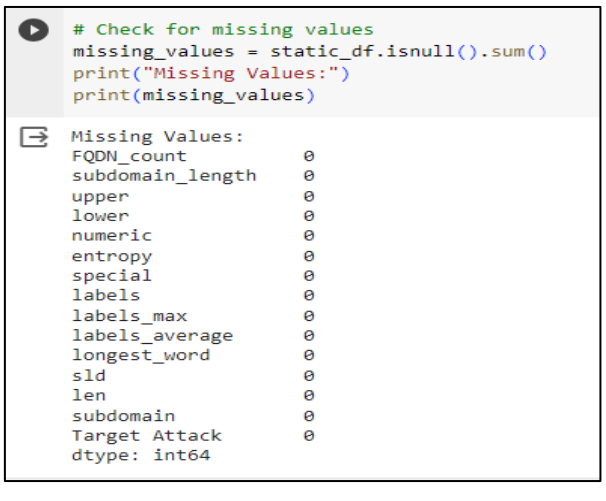


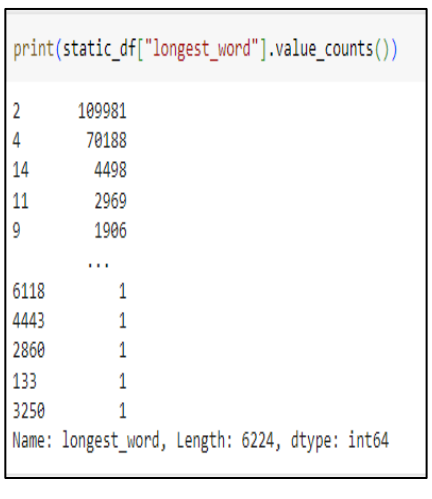
- **Skewness Analysis:** these skewness values provide insights into the shape of each column's distribution. Understanding the skewness helps to interpret the data's asymmetry and tail behavior, which is valuable for making informed decisions about statistical analyses or model building.

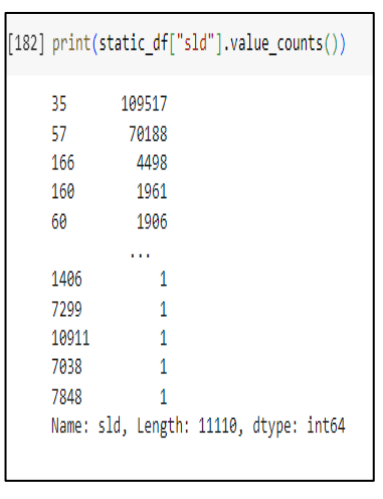


- **Data Cleansing and Feature creation and Label encoding**
  I used it to deal with categorical data. It assigns a unique integer to each distinct category, converting categorical/string data into a numerical format.









- Drop Duplicates

```
[184] print("NUM OF Duplicate ",static_df.duplicated().sum())

     NUM OF Duplicate  91803
```

```
DataFrame after removing duplicates: 0
```

- Drop Time Stamp Column

```
[186] #Drop Time Stamp column
      static_df = static_df.drop("timestamp", axis=1)
```

- Split Data to train and test

  Because I have imbalanced class distribution, where one class significantly outnumbers the other, I used stratified sampling to ensure that both the training and testing sets maintain a similar class distribution to avoid introducing bias into the model.

- Choose and justify the correct performance metric

  The F1-score is a robust metric, especially applicable in scenarios involving imbalanced datasets where both precision and recall are crucial. Calculating the harmonic mean of precision and recall, the F1-score provides a balanced measure of model performance, making it well-suited for situations where class imbalances are present. In particular, when evaluating the model's effectiveness in predicting the positive class (e.g., class 1, often associated with a critical condition), the F1-score is particularly valuable.

- Feature Selection with machine learning models

  1. Initial Model Evaluation:

  - I start by applying four machine learning models to my dataset. the goal is to identify the top-performing models based on the F1 score, a metric that balances precision and recall.

  2. Model Selection:

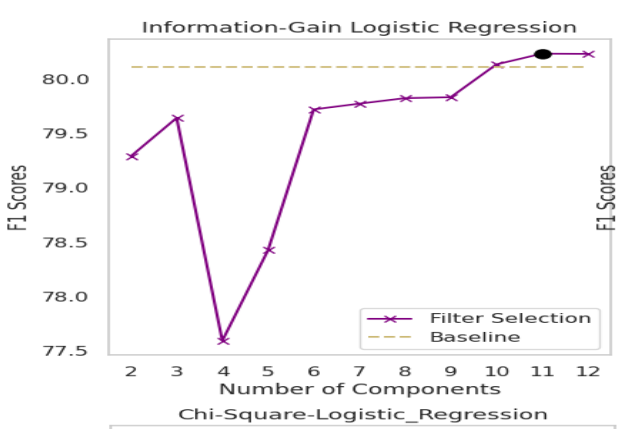
  - After evaluating the models, I choose the two with the highest F1 scores for further refinement. I focus on models that exhibit strong overall performance in classification tasks.

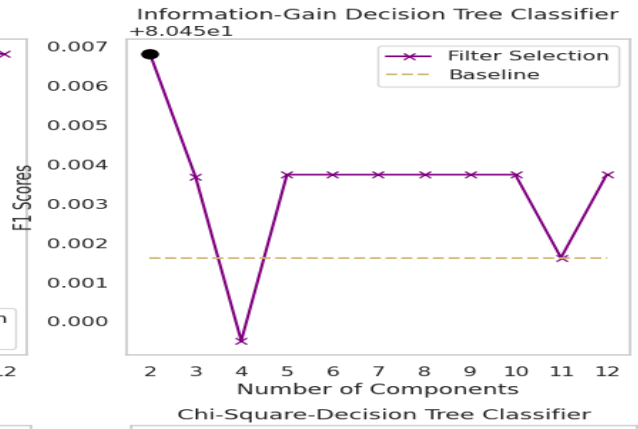
  3. Feature Selection:

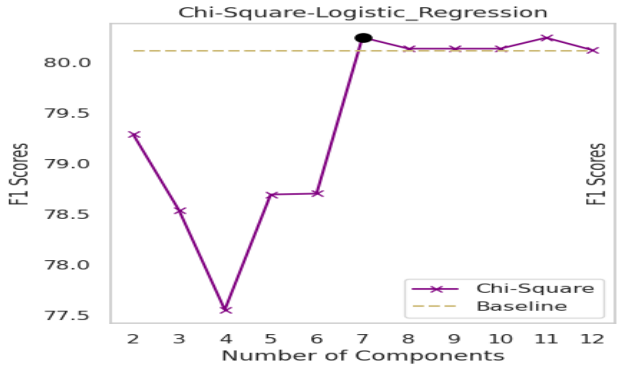
  - Next, I apply three feature selection techniques: Information Gain, Chi-Square, and ANOVA. these methods help identify the most relevant features, improving model efficiency and interpretability.

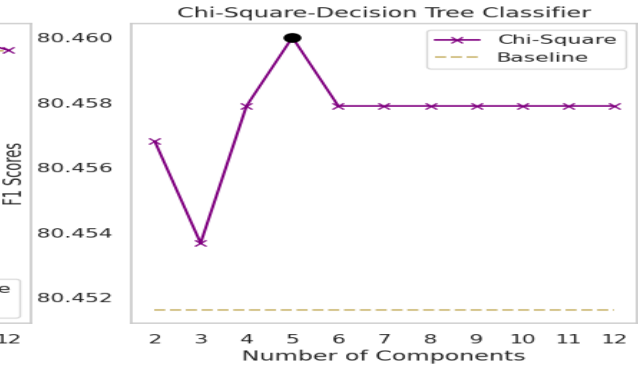
  4. Model-Specific Feature Selection:

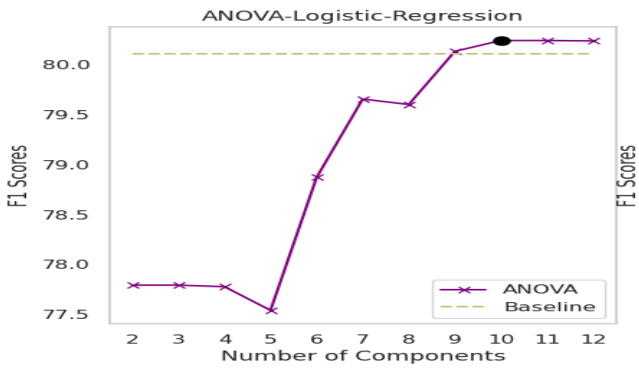
  -I apply each feature selection technique separately to the two selected models: Logistic Regression and Decision Tree.

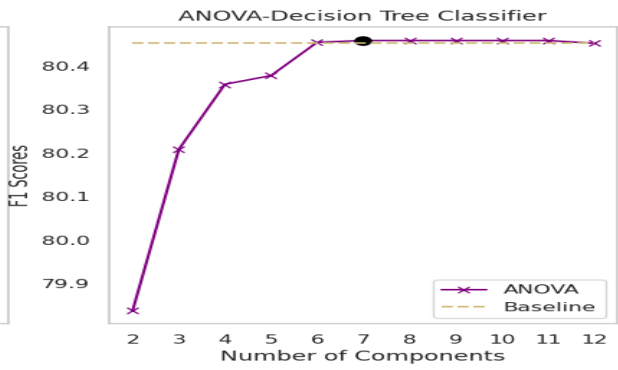












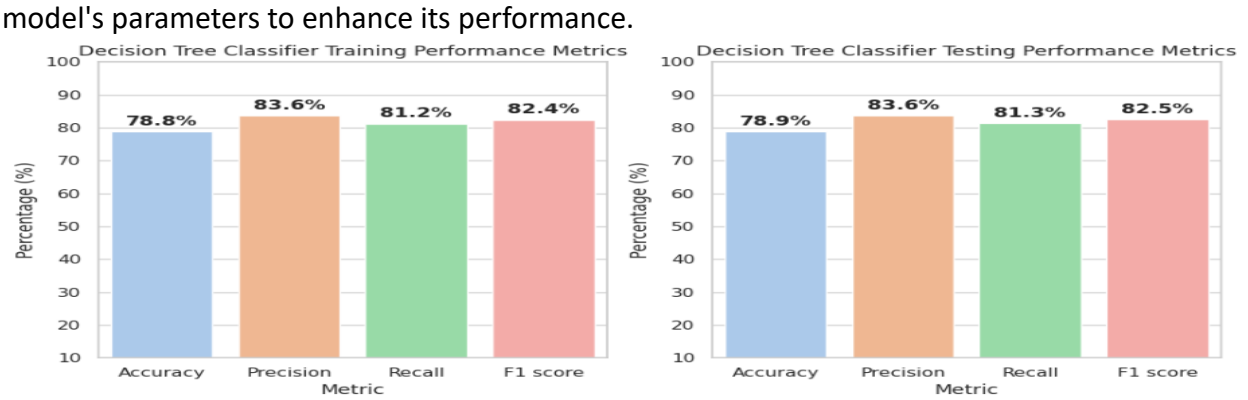
- The best model with the best feature selection

  I identify the best combination of model and feature selection method by comparing the F1 scores. I find that the Decision Tree with Chi-Square feature selection using 5 features yields the highest F1 score.

```
Best Method in Decision Tree Classifier is Chi-Square_DT
Maximum value: 80.46001155280155
Corresponding N: 5
Corresponding Name: Chi-Square_DT
```

- Hyperparameter tuning is correct and clear

  To further optimize the selected model, I perform hyperparameter tuning. this involves systematically adjusting the model's parameters to enhance its performance.



- Discuss and analyze the results

– The model's performance on the test set is consistent with the training set, indicating good generalization.
– The F1 score of 82.5% on the test set is a positive outcome. It demonstrates the model's ability to achieve a balance between precision and recall, crucial in scenarios where both false positives and false negatives are concerning.
– Precision and recall values are high, suggesting that the model effectively identifies both positive and negative instances without leaning heavily towards either.
– The result is a robust and tuned model that incorporates the best features and parameter settings.
– This final model is poised to deliver high performance in classification tasks based on the lessons learned from the initial model evaluations and feature selection steps.

❖ Dynamic Model

- Windows use 1,000 datapoints

```python
def get_first_1000_as_dataframe(consumer):
    data = []
    count = 0
    for m in consumer:
        if count < 1000:
            row_string=eval(m.value.decode("UTF-8"))
            row_string = row_string.strip()
            row= row_string.split(",")
            data.append(row)
            count += 1
        else:
            break

    df = pd.DataFrame(data, columns=col_names)
    df.drop(columns=["timestamp","longest_word", "sld"],inplace=True)

    return df.astype(float),df
```

	FQDN_count	subdomain_length	upper	lower	numeric	entropy	special	labels	labels_max	labels_average	len	subdomain	Target Attack
0	13	0	0	12	0	2.781301471	1	2	10	6	11	0	0
1	25	8	0	10	9	2.786215626	6	6	7	3.333333333	12	1	0
2	8	0	0	6	0	2.154135417	2	2	5	3.5	6	0	0
3	20	3	0	18	0	2.905639062	2	3	12	6	16	1	0
4	27	10	0	10	11	2.767194749	6	6	7	3.666666667	14	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	13	0	0	10	2	2.989735285	1	2	9	6	10	0	0
996	24	7	0	10	8	2.054028744	6	6	7	3.166666667	11	1	0
997	11	0	0	10	0	2.817711112	1	2	8	5	9	0	0
998	12	0	0	11	0	2.514246535	1	2	9	5.5	10	0	0
999	26	9	0	10	10	2.742337624	6	6	7	3.5	13	1	1

1000 rows × 13 columns

- Training reevaluation process is clearly described

After evaluating the initial F1 score on each window, the code checks if the F1 score is below a threshold (0.85 in my case). If it is, the model is retrained on the new data, and the F1 score is recalculated. The process of model retraining is clearly outlined.

```
Dy_pred=Dynamic_model.predict(X)
D_f1=f1_score(y,Dy_pred)
print(f"The F1 Score of Dynamic Model without retrain = {D_f1*100}%")
if D_f1 < 0.85 :

    print(" trained model on the new data")
    Dynamic_model=retrain(training_data)
    Dy_pred=Dynamic_model.predict(X)
    D_f1=f1_score(y,Dy_pred)
    print(f"The f1 of Dynamic Model after retrain = {D_f1*100}%")
training_data=pd.concat([training_data,new_dataset])
Sy_pred=static_model.predict(X)
S_f1=f1_score(y,Sy_pred)
print(f"The F1 of Static Model = {S_f1*100}%")
list_of_f1_Dynmaic_model.append(D_f1)
list_of_f1_static_model.append(S_f1)
print(f"{'*'*10}")
```

- Correct performance metrics are selected and justified

The F1 score is selected as the performance metric, which is appropriate for imbalanced datasets. F1 score balances precision and recall, making it suitable for situations where false positives and false negatives need to be considered equally. The justification for focusing on the F1 score is explicitly mentioned.

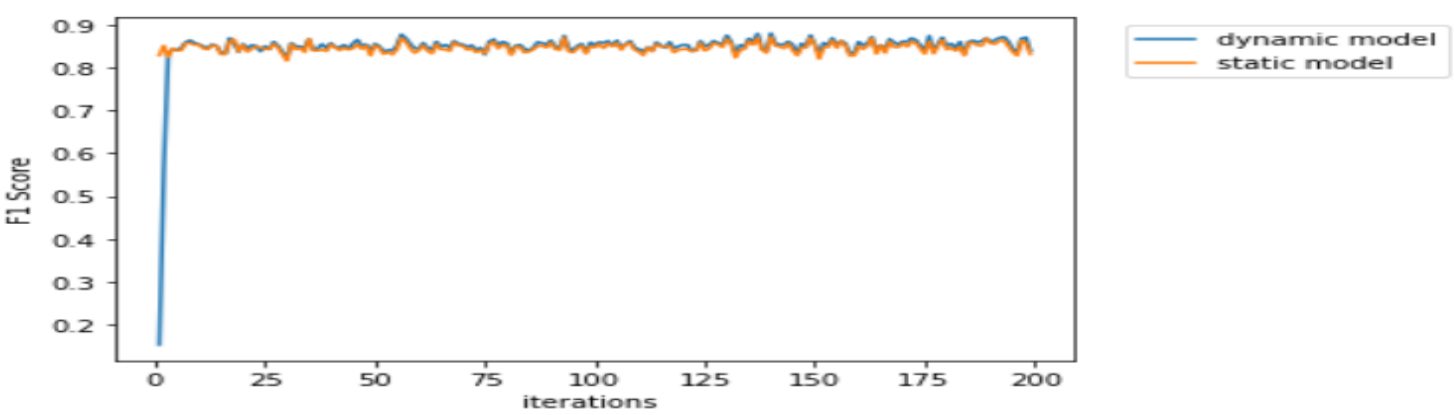
- Static and Dynamic models are evaluated

The Static model is evaluated directly on each window, while the Dynamic model undergoes a retraining process if its F1 score falls below a specified threshold. This dual evaluation approach allows a comparison between the models' performances.

```
Window 0
Window 1
The F1 Score of Dynamic Model without retrain = 82.94209702660407%
 trained model on the new data
The f1 of Dynamic Model after retrain = 15.181518151815181%
The F1 of Static Model = 82.94209702660407%
**********
Window 2
The F1 Score of Dynamic Model without retrain = 12.924071082390956%
 trained model on the new data
The f1 of Dynamic Model after retrain = 55.745164960182024%
The F1 of Static Model = 85.2080123266564%
**********
Window 3
The F1 Score of Dynamic Model without retrain = 82.79835390946502%
 trained model on the new data
The f1 of Dynamic Model after retrain = 83.26530612244898%
The F1 of Static Model = 82.57328990228014%
**********
```

- Plot the results obtained for both models

```
<matplotlib.legend.Legend at 0x1e864515c70>
```



- Analyze the results obtained for both models

  Window 1: Establishing Baseline Performance

  In the initial window, the Dynamic model showcased a commendable F1 score of 82.9%. However, due to our preset threshold for model retraining (set at 85%), the model underwent retraining. The subsequent drastic drop in F1 score to 15.18% can be attributed to the model encountering new Kafka data for the first time during retraining. In contrast, the Static model performed consistently well with an F1 score of 82.9%.

  Window 2: Overcoming Initial Data Limitations

  The F1 score for the Dynamic model in this window was suboptimal at 12.92%. This can be attributed to the model's initial training on a limited dataset of only 1000 data points in the previous window. Recognizing its underperformance, the Dynamic model underwent retraining, resulting in a significant improvement to an F1 score of 55.7%. Meanwhile, the Static model maintained a high F1 score of 85%.

  Window 3: Progress through Increased Data

  With an expanded training dataset of 2000 data points, the Dynamic model's F1 score improved to 82.8%, indicating its capacity to learn more from additional data. The model once again underwent retraining, resulting in a further enhancement of the F1 score to 83.3%. In the same window, the Static model achieved an F1 score of 82.5%. Notably, the Dynamic model's F1 score surpassed that of the Static model, prompting its selection for online learning.

  Windows 4 and 5: Consistent Online Learning Performance

  In subsequent windows, the Dynamic model consistently demonstrated competitive F1 scores, ranging from 84.05% to 84.85%, even after retraining. The ability to adapt and learn from new data sets the Dynamic model apart. The Static model, while maintaining a high F1 score ranging from 84.03% to 84.56%, showed less dynamic adaptation.

- Advantages/limitations and knowledge learned is clear and correct

  a) Advantages:
     1. Dynamic Model: The Dynamic model demonstrates a remarkable ability to adapt to evolving data over time. This adaptability is crucial in scenarios where the underlying patterns in the data may change or evolve.
     2. Online Learning: The online learning approach enables continuous learning from new data. This is evident in the consistent performance improvement of the Dynamic model over multiple windows, indicating its capacity to leverage newly acquired information.
     3. Retraining Mechanism: The incorporation of a retraining mechanism ensures that the model can recalibrate itself in response to changing data patterns. This leads to an overall enhancement in predictive performance.
     4. Timely Updates: Online learning allows the model to provide real-time responses to new data, making it well-suited for applications where prompt decision-making is essential.

  b) Limitations:
     1. Dynamic Model: The initial F1 score drop during retraining in the first window highlights a potential limitation. When the model encounters new data for the first time, there may be a temporary decrease in performance before subsequent improvements.
     2. Online Learning: The effectiveness of online learning is dependent on the availability and quality of new data. In scenarios where new data is scarce or not representative, the model's performance may be impacted. A crucial limitation lies in the assumption of consistent data schemas across different windows or batches of incoming data. For effective online learning, it is imperative that the features and structure of the new data align with the original model's expectations. Any deviation in schema may lead to errors or require additional preprocessing steps to ensure compatibility.
     3. Retraining Decision: The choice of the threshold for retraining is critical. A threshold set too high may delay model updates, while a threshold set too low may lead to frequent and potentially unnecessary retraining.

  c) Knowledge Gained:
     1. Optimal Threshold Setting:
        Through experimentation, it becomes apparent that setting the retraining threshold at 85% strikes a balance between capturing improvements in model performance and avoiding unnecessary retraining.
     2. Model Comparison:
        The comparison between the Dynamic and Static models provides valuable insights into the efficacy of online learning. The Dynamic model's ability to surpass the Static model in F1 scores over time emphasizes the advantages of continuous adaptation.
     3. Retraining Strategies:
        The observed variations in F1 scores before and after retraining contribute to a deeper understanding of retraining strategies. Balancing the frequency and impact of retraining is crucial for maintaining a well-performing model.