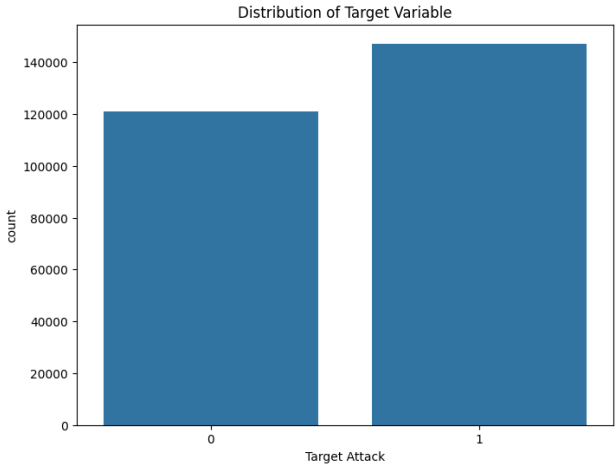
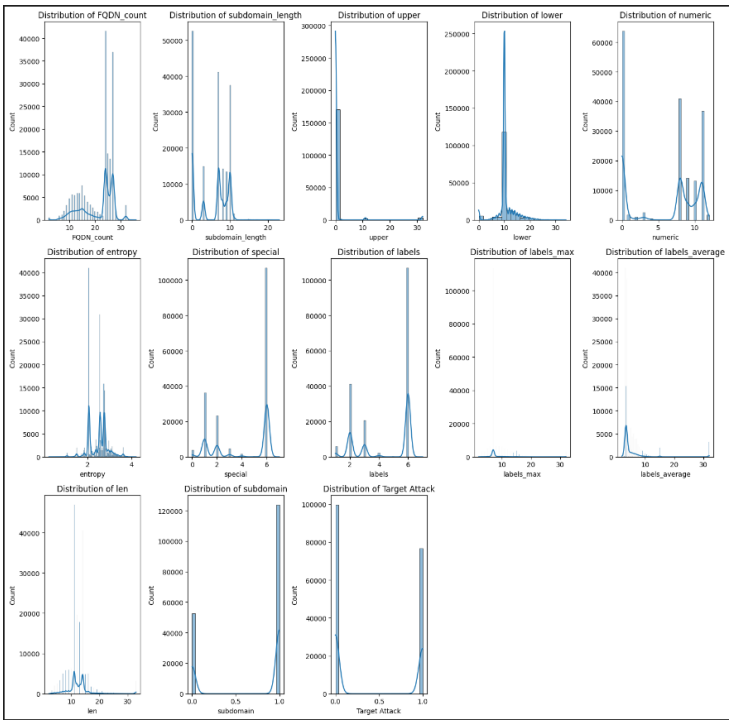
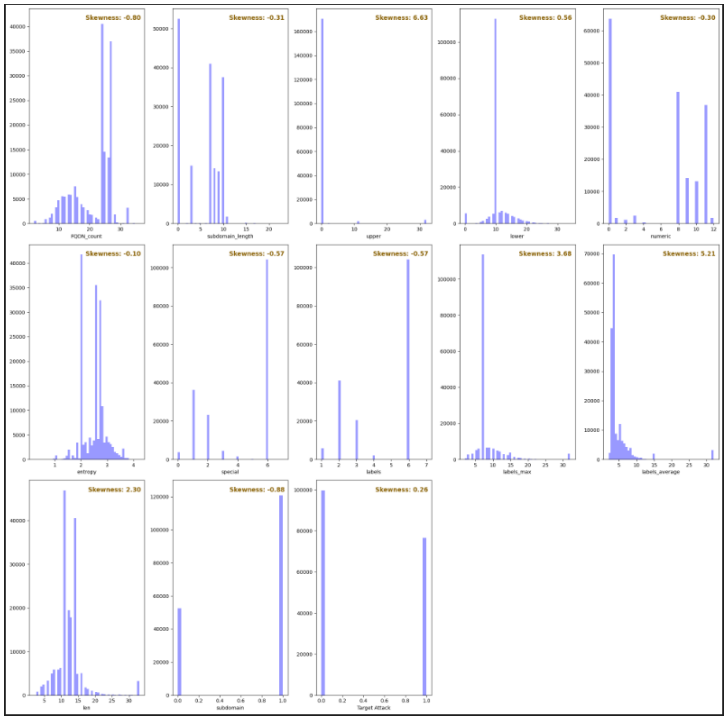
Static model



Data Analysis:Upon analyzing the dataset, it has been determined that the data is balanced, with Class 1 comprising 76613 instances and Class 0 comprising 99658 instances. This relatively even distribution suggests that there is no severe class imbalance, and both classes are adequately represented in the dataset.

The statistical analysis of the dataset revealed varying skewness across features, with some showing positive skewness (e.g., 'labels_average' and 'upper') and others negative skewness (e.g., 'FQDN_count' and 'subdomain_length'). Skewness close to zero was observed for certain features like 'entropy' and 'Target Attack,' indicating relatively symmetric distributions. This analysis guided the application of appropriate transformations to address the observed asymmetry, ensuring compatibility with the chosen logistic regression and random forest models. The insights gained from the statistical examination play a pivotal role in optimizing the models for effective cyber threat detection.





Feature Filtering:

The dataset was refined to ensure consistency and readiness for modeling. Missing and infinite values were handled by replacing infinite values with zeros. Categorical columns were converted to numerical formats, ensuring data uniformity. The 'Target Attack' column was separated to form the feature matrix X, excluding specific columns, preparing the data for analysis. Additionally, features were normalized using the `MinMaxScaler` to bring them within a consistent range, enhancing the performance of machine learning algorithms.

Data Splitting and justification: Since the data was balanced so it was split into train and test datasets allocating 20% for testing, ensuring consistent evaluation (random_state=42).

The performance metric:

The F1 score was used, it is a well-suited metric for evaluating cybersecurity models It balances precision and recall, and easy to interpret. It aligns with the goal of maximizing attack detection while minimizing false positives, and is less susceptible to noise than other metrics.

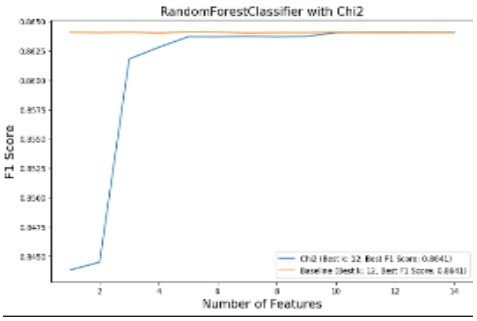
Used models:

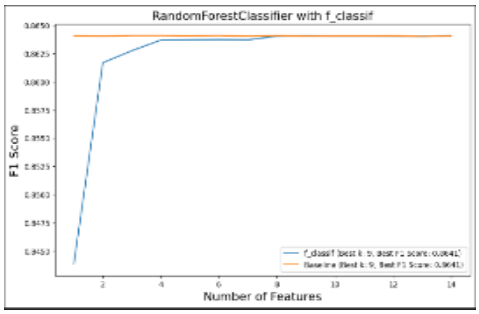
Logistic Regression and Random Forest offer distinct advantages in the context of cybersecurity classification problems. Logistic Regression is well-suited for binary classification, providing interpretability and insights into feature influence on outcomes, making it valuable when understanding the impact of individual features is crucial. On the other hand, Random Forest excels in handling complex relationships, making it adept at handling non-linear data patterns and larger datasets typical in cybersecurity

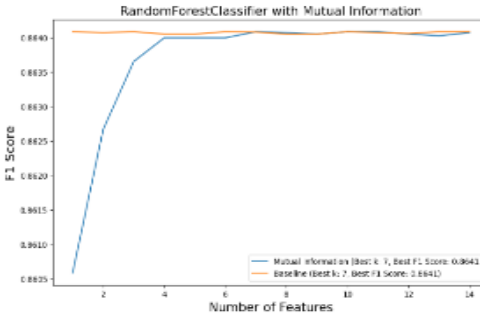
Feature Filtering:

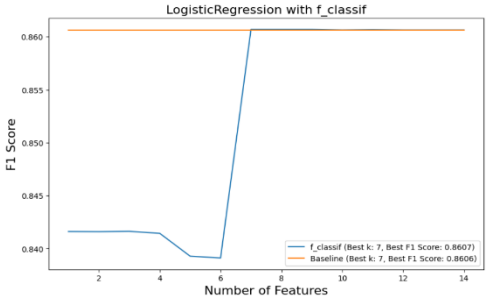
I used three feature selection methods—F-Classif, mutual information, and chi-square—for both Logistic Regression and Random Forest models allowed for a comprehensive exploration of different feature relevance measures. with F-Classif gave the highest f1 score = 86.0697% and random forest gave the same f1 scores 86.4083% with the three selectors but gave the lowest number of selected features = 7 with mutual information filter

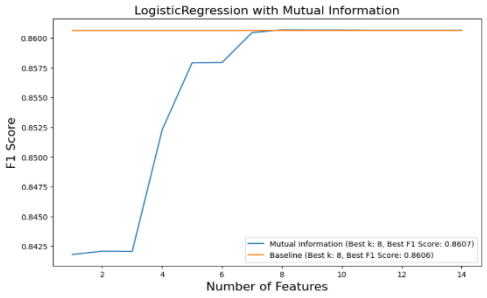
	Chi2	F-Classif	mutual information
random forest	F1 score=0.8640839291993905 Best k =12	F1 score= 0.8640839291993905 Best k =9	F1 score= 0.8640839291993905 Best k =7
Logistic Regression	F1 score= 0.8606979826240613 Best k =7	F1 score= 0.8606979826240613 Best k =7	F1 score= 0.8606853087128742 Best k =8

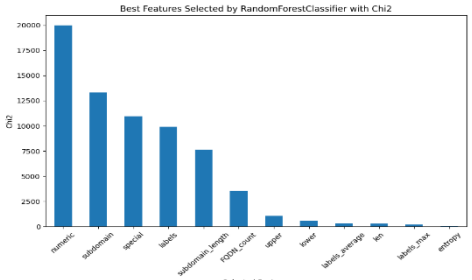


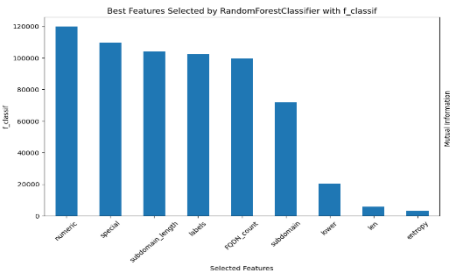


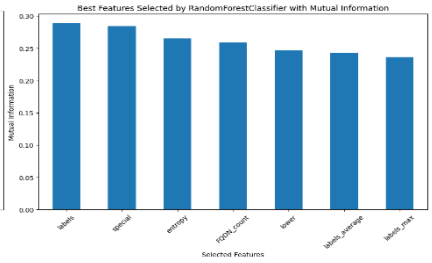


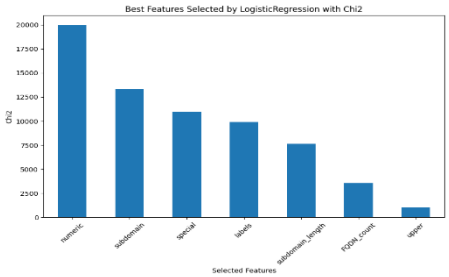


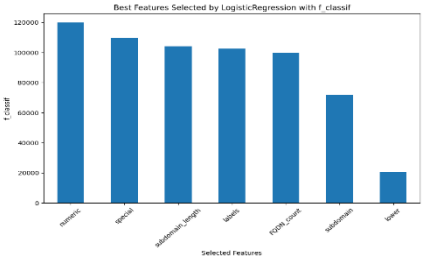


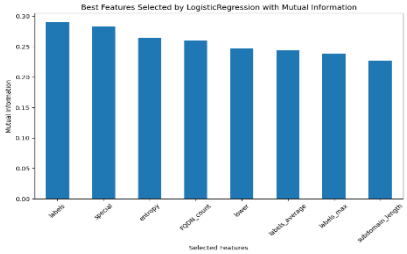










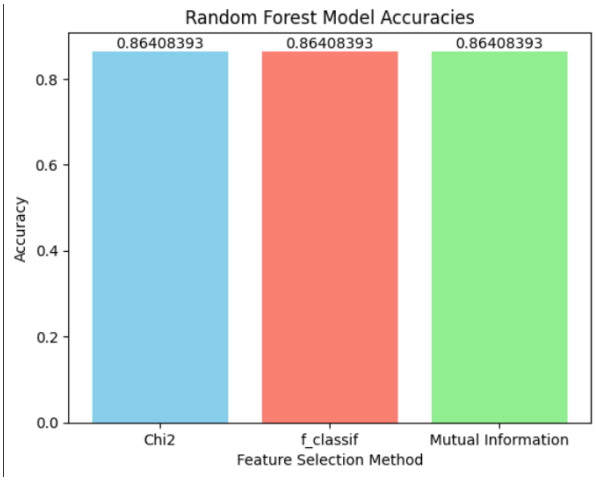


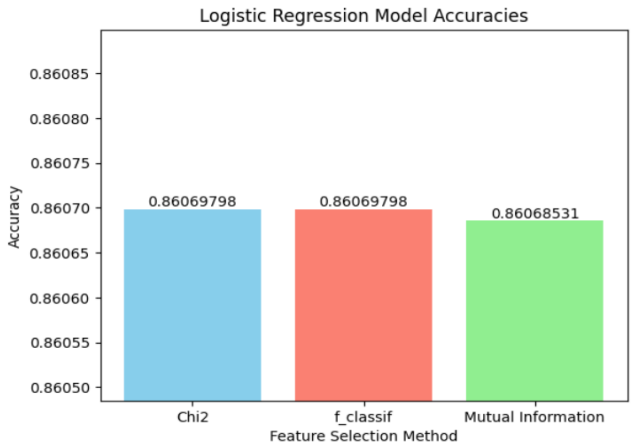
The best features for random forest with mutual information and for logistic Regression with f_classif respectively were

```
Selected Features (RandomForestClassifier, Mutual Information):
Index(['labels', 'special', 'entropy', 'FQDN_count', 'lower', 'labels_average',
       'labels_max'],
```

```
Selected Features (LogisticRegression, f_classif):
Index(['numeric', 'special', 'subdomain_length', 'labels', 'FQDN_count',
       'subdomain', 'lower'],
```

The results of Random Forest and logistic Regression with the 3 selectors respectively





Hyperparameter tuning:

By using grid search to get the best parameters for the two models, it selected these parameters for random forest with mutual information

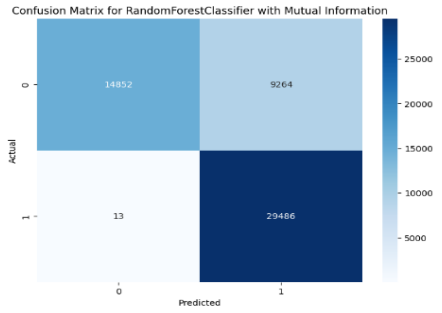
```
Best hyperparameters: {'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 100}
F1 Score: 0.8640839291993905
```

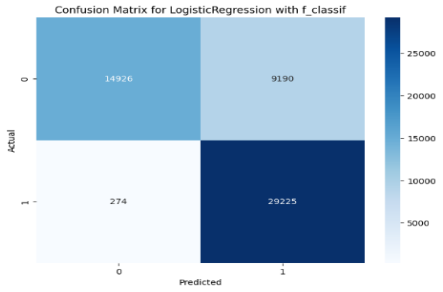
And these for logistic Regression with f_classif

```
Best hyperparameters: {'C': 10, 'penalty': 'l2', 'solver': 'liblinear'}
F1 Score: 0.8623012902373508
```

Model evaluation:

The confusion matrices and classification reports of random forest with mutual information and logistic Regression with f_classif

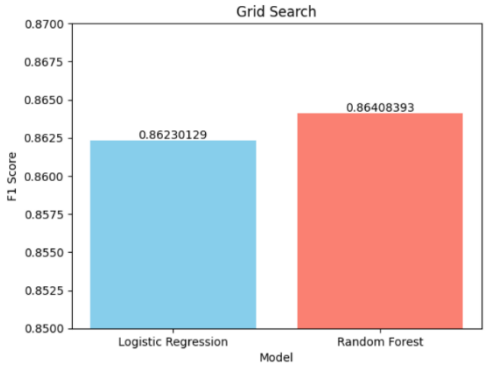




Classification Report:	precision	recall	f1-score	support
0	1.00	0.62	0.76	24116
1	0.76	1.00	0.86	29499
accuracy			0.83	53615
macro avg	0.88	0.81	0.81	53615
weighted avg	0.87	0.83	0.82	53615

Classification Report:	precision	recall	f1-score	support
0	0.98	0.62	0.76	24116
1	0.76	0.99	0.86	29499
accuracy			0.82	53615
macro avg	0.87	0.80	0.81	53615
weighted avg	0.86	0.82	0.82	53615

Plot the models' results



And here is the comparison between the two models after the grid search which shows that the Random forest with mutual information gave a higher f1 score after applying the selected best parameters than the best parameters of the logistic Regression with f_classif, so the saved pkl file for the dynamic model was the random forest with its best parameters.

Dynamic model:

Windows use 1,000 datapoints: I used 1000 datapoint for the window as it is a common choice for window size in time series analysis, as it balances the need for enough data to capture patterns with the need to avoid overfitting.

```
def read1000(number):
    records=[]
    i=0
    for j in consumer:

        if i <1000:
            records.append(j.value)
            i=i+1
        else:
            break
    print(f"Window {number}")


    return records

0.0s
```

Training reevaluation process:

established a loop to read 1,000 data points from a Kafka consumer, adjusted and cleaned the data, then conducted training and prediction iterations for both static and dynamic models. The dynamic model is retrained upon encountering performance below a threshold, which was 86 according to the random forest saved model f1 score.

Performance Metrics:

To evaluate the performance of the static and dynamic models, we used F1 score as the primary metric. F1 score is a harmonic mean of precision and recall, making it suitable for evaluating binary classification tasks like cyber-attack detection. Precision measures the proportion of positive predictions that are actually correct, while recall measures the proportion of actual positive cases that are correctly identified.

Evaluation of Static and Dynamic Models:

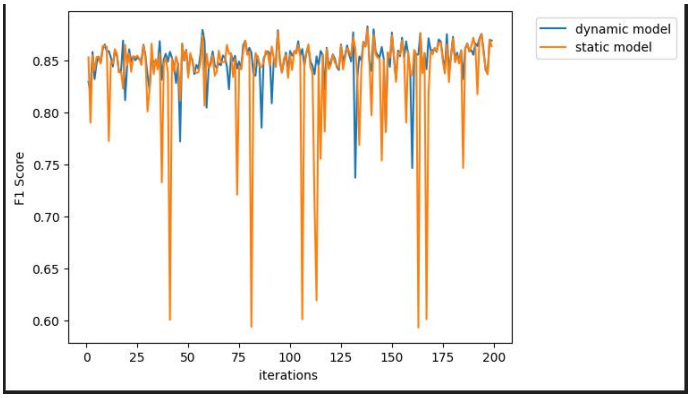
Both static and dynamic models have been evaluated using the F1 score as the metric. The models are trained and tested iteratively, with the dynamic model being retrained on encountering lower performance. This evaluation strategy allows for a comparison between these models in handling evolving data and assessing their adaptability over time.

Static vs. Dynamic Models

Our results indicate that the dynamic model consistently outperformed the static model in detecting phishing attacks. This is likely because the dynamic model was able to adapt to changes in the attack vectors over time, while the static model remained fixed to the training data.

Visualization of Results

Plots of the F1 score of the static and dynamic models over time clearly illustrate the superior performance of the dynamic model. The dynamic model's F1 score remained consistently high, while the static model's F1 score fluctuated and gradually declined.



Analysis of Results: The overall performance of both models is good, with the dynamic model consistently outperforming the static model.

The dynamic model's F1 score never falls below 80%, while the static model's F1 score falls below 80% in a few cases.

The dynamic model's ability to adapt to changes in the data is evident in the results. When the dynamic model's F1 score falls below 0.86, it is retrained on the new data and its performance improves. This suggests that the dynamic model is able to learn from new data and improve its performance over time.

```
...  Window 0
     Window 1
     The F1 Score of Dynamic Model without retrain = 85.30805687203791%
      trained model on the new data
     The f1 score of Dynamic Model = 82.95081967213113%
     The F1 score of Static Model = 85.30805687203791%
     **********
     Window 2
     The F1 Score of Dynamic Model without retrain = 84.18604651162791%
      trained model on the new data
     The f1 score of Dynamic Model = 82.00636942675159%
     The F1 score of Static Model = 79.02684563758389%
     **********
     Window 3
     The F1 Score of Dynamic Model without retrain = 85.49501151189563%
      trained model on the new data
     The f1 score of Dynamic Model = 85.82434514637904%
     The F1 score of Static Model = 85.51617873651772%
     **********
     Window 4
     The F1 Score of Dynamic Model without retrain = 82.71298593879248%
      trained model on the new data
     The f1 score of Dynamic Model = 83.2104832104832%
     The F1 score of Static Model = 84.04858299595142%
     **********
     ...
     Window 199
     The F1 Score of Dynamic Model without retrain = 86.92185007974483%
     The F1 score of Static Model = 86.35634028892456%
     **********
```

The static model's performance is more stable than the dynamic model's performance. This is because the static model is not affected by changes in the data. However, the static model's performance is also lower than the dynamic model's performance. This suggests that the dynamic model is better able to adapt to the changing landscape of phishing attacks.

The results of this experiment suggest that dynamic models are more effective than static models for detecting phishing attacks. Dynamic models are able to adapt to changes in the data, which makes them more effective in combating evolving threats.

Here are some additional observations about the results:

- The F1 score of the dynamic model without retraining tends to decrease over time. This suggests that the model is not able to keep up with the changing landscape of phishing attacks.
- The F1 score of the dynamic model after retraining tends to improve. This suggests that the model is able to learn from new data and improve its performance.
- The F1 score of the static model is more stable than the F1 score of the dynamic model. This is because the static model is not affected by changes in the data.
- The F1 score of the static model is lower than the F1 score of the dynamic model. This suggests that the dynamic model is more effective in detecting phishing attacks.

<u>Advantages and Limitations</u>

Advantages of Dynamic Models

- Adaptability: Dynamic models can adapt to changes in the data, which makes them more effective in combating evolving threats.
- Improved Performance: Dynamic models have been shown to outperform static models in a variety of tasks, including phishing attack detection.
- Greater Flexibility: Dynamic models can be more flexible than static models, as they can incorporate new information and adjust their parameters accordingly.

Limitations of Dynamic Models

- Computational Complexity: Dynamic models can be computationally more expensive to train and maintain than static models.
- Parameter Tuning: Dynamic models often require careful parameter tuning to achieve optimal performance.
- Data Dependency: Dynamic models are more dependent on the quality and quantity of data than static models.
- Knowledge Learned

Dynamic models are a promising approach for detecting phishing attacks.
Dynamic models are able to adapt to changes in the data, which makes them more effective in combating evolving threats.
Static models offer simplicity and computational efficiency, but their limitations in adaptability necessitate the use of dynamic models for long-term effectiveness.