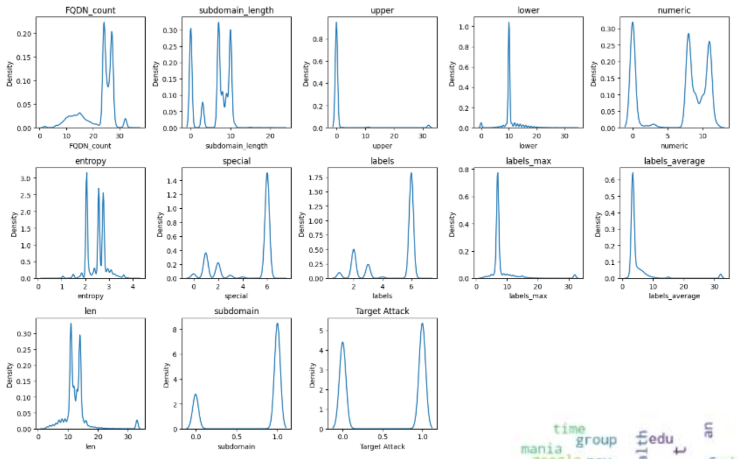
1. Static Model

- **Statistical analysis of data**

  I identified columns with object data type, duplicated rows, the number of null values in each column, skewness, created word clouds for object type features, and plotted KDE plots for further examine of the features.



Word Cloud of SLD

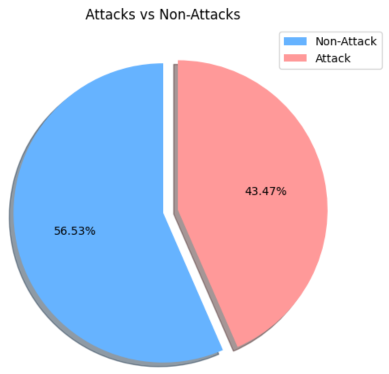
Word Cloud of Longest Word

- **Data Cleansing and Feature creation**

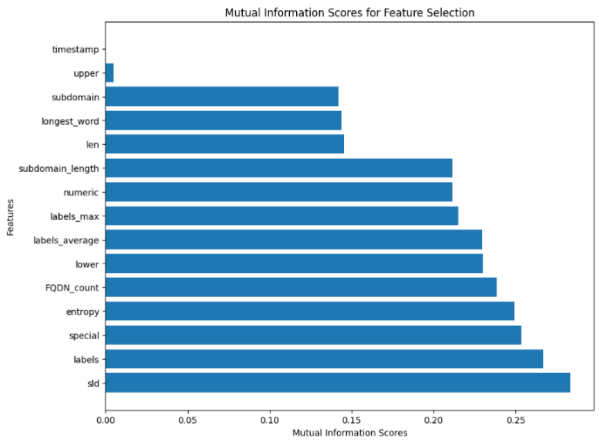
  I removed duplicated rows and null values. Building on insights gained from the statistical analysis, I transformed the 'longest_word' column by representing string values as their length while keeping numerical values. Additionally, I modified the 'sld' column by replacing its values with their frequencies.

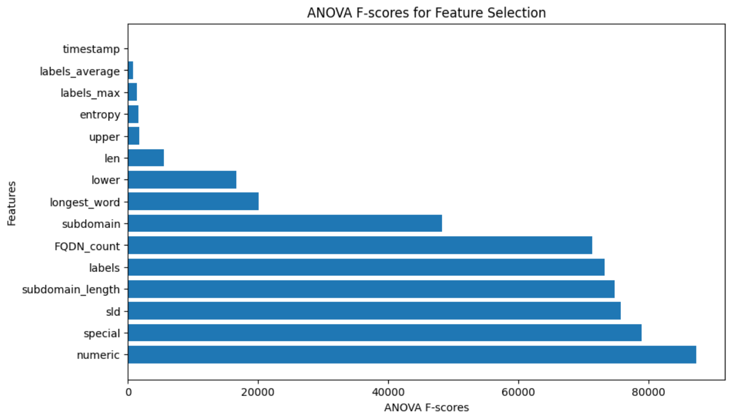
- **data imbalance**

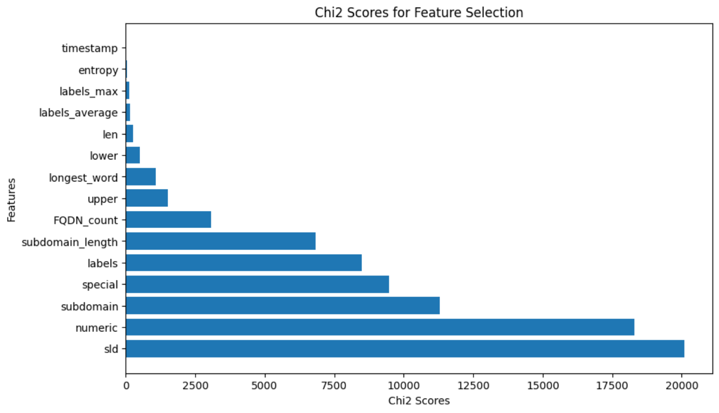
  By analyzing the target values, it was observed that the distribution between Attacks and Non-Attacks was 43.47% to 56.53%



- **Feature Selection**
  I used three different methods: Mutual Information, ANOVA, and Chi-squared. I visualized the features sorted by their importance and saved the top 8 features from each method.





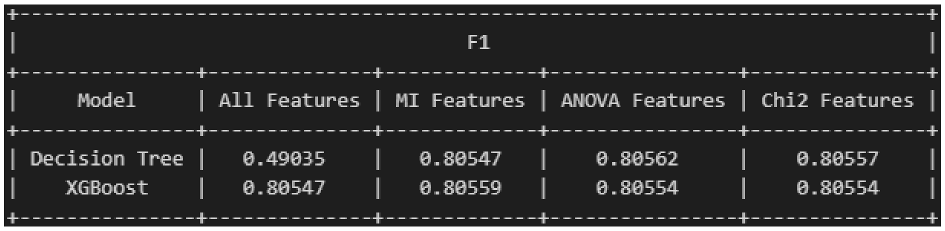


- **Data Splitting**
  The data was divided into training and testing datasets, with an 80% to 20% ratio, respectively. I used stratification to ensure proper distribution in both sets.

- **performance metrics**
  Given that the primary objective is to classify attacks, with the cost of missing an attack being significantly higher than classifying benign activity as attacks, I initially considered using recall as the main performance metric. However, all models exhibited a very high recall score of over 99.9%. Therefore, I used the F1-score as the main performance metric, as it represents the harmonic mean between recall and precision.

- **Model Selection and comparison**
  I chose Decision tree and XGBoost for their distinct characteristics. Decision trees are easy to interpret, have no assumptions about data distribution, and can handle non-linearities. However, Decision trees are prone to overfitting, and small changes in data can cause significant changes in the resulting tree structure making it somewhat unstable. XGB has an improved performance as an ensemble method that builds multiple decision trees. It uses regularization terms to deal with overfitting. However, it's more complex making it harder to interpret and requires careful hyperparameters tuning.

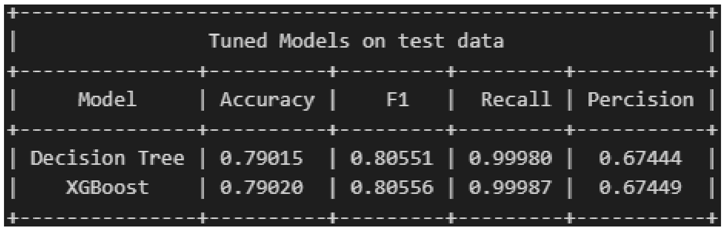
  I compared the performance of both models on: all the features, Mutual Information features, ANOVA features, and Chi-squared features. The result showed that the Decision Tree had the highest performance using the ANOVA features while XGB had the highest performance using the Mutual Information features.



```
+-------------------------------------------------------------------------+
|                                    F1                                   |
+---------------+-------------+------------+---------------+--------------+
|     Model     | All Features | MI Features | ANOVA Features | Chi2 Features |
+---------------+-------------+------------+---------------+--------------+
| Decision Tree |   0.49035   |   0.80547  |    0.80562    |    0.80557   |
|    XGBoost    |   0.80547   |   0.80559  |    0.80554    |    0.80554   |
+---------------+-------------+------------+---------------+--------------+
```

- **Hyperparameter tuning**
  I fine-tuned the Decision tree using ANOVA features and XGB using Mutual Information features and compared between them. The evaluation was performed on the test data and recorded all metrics for a deeper analysis.

- **Results Discussion and Analysis**
  The Decision Tree showed a higher performance on the training data compared to XGB. However, XGB outperformed the Decision Tree, showing a higher performance on the testing data indicating that it generalizes better to unseen data. The champion model is XGB, and the best features are the 8 features selected by the Mutual Information method.



```
+-------------------------------------------------------------+
|                   Tuned Models on test data                 |
+---------------+----------+---------+---------+-----------+
|     Model     | Accuracy |    F1   |  Recall | Percision |
+---------------+----------+---------+---------+-----------+
| Decision Tree | 0.79015  | 0.80551 | 0.99980 |  0.67444  |
|    XGBoost    | 0.79020  | 0.80556 | 0.99987 |  0.67449  |
+---------------+----------+---------+---------+-----------+
```

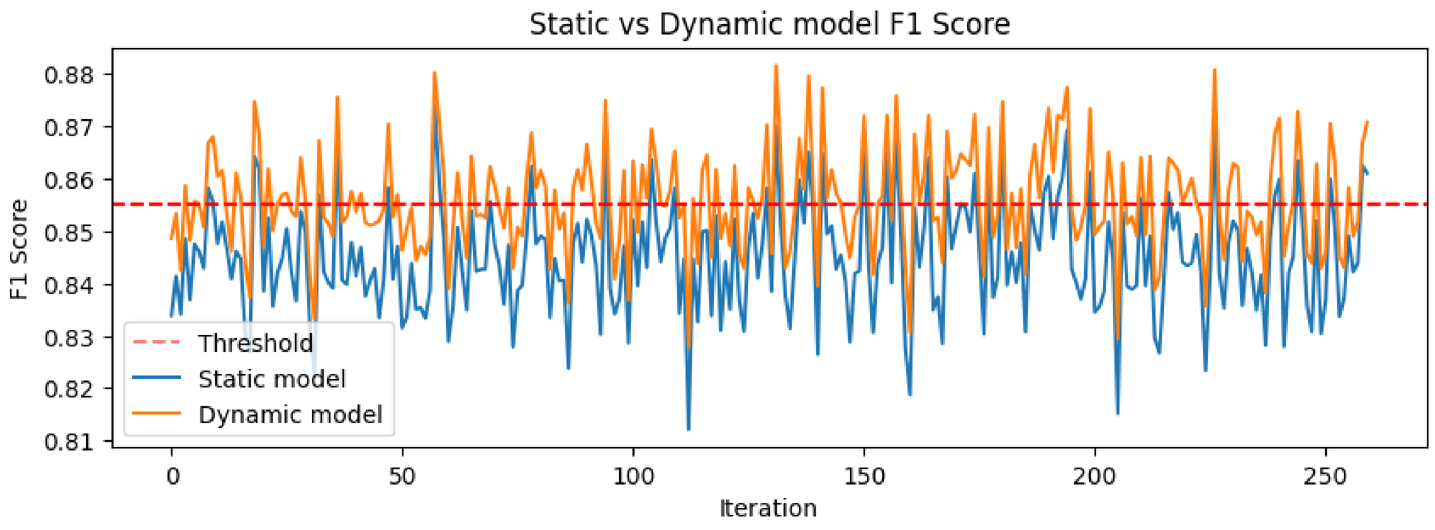
2. Dynamic model

- **Windows of 1000 datapoints**
  The data was passed through multiple functions to extract 1000 observations, followed by cleaning and preprocessing steps similar to those applied in the static part. The data was then scaled using the same scaler that was fitted in the Static part.

- **Performance metrics**
  I used F1-score as the main performance metric in the Dynamic model, consistent with the Static Part.

- **Evaluation and Retraining**
  Both Static and Dynamic models are continuously evaluated on each window of the data, and the results are appropriately saved for later analysis and visualization. I set an F1-score threshold for the dynamic model triggering the retraining process if the performance falls below this threshold. the model is then updated after fitting on this window, facilitating incremental learning. This strategy ensures that the model is trained on new subsets of data where it underperformed, without compromising the original training on the older data.

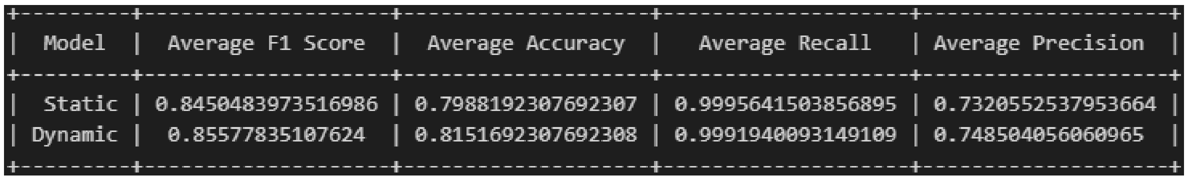
- **Results Discussion and Analysis**
  Since the F1-scores of the Dynamic model are stored after retraining, both models don't start from the same point as they started below the threshold which retrained the Dynamic model and increased its performance. As seen in the fig comparing the results of both models, the Dynamic model is always outperforming the Static model particularly at points below the threshold. This showcases the success of the retraining process in improving the model's performance.



- **Advantages and Limitations**

  The advantages of the Dynamic model over the Static model include:

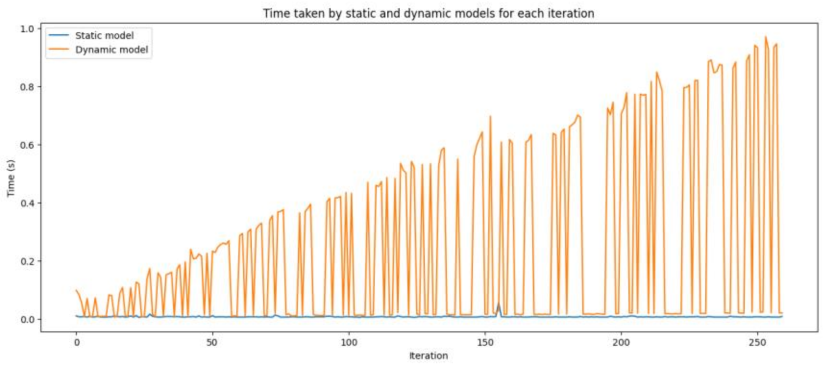
  - **Continuous Improving:** The Dynamic model continuously improves its performance and adapts to new data, as opposed to being only fitted to historical data.
  - **Incremental Learning:** Using the continues learning capabilities in XGB enables the adaptation to streaming data without the need to refit the model on the entire dataset. This feature conserves valuable time and resources.
  - **Better Performance:** The Dynamic model consistently outperforms the Static model across all metrics, except for a marginal decrease in recall compared to the Static model.



```
+---------+--------------------+--------------------+--------------------+--------------------+
|  Model  |  Average F1 Score  |  Average Accuracy  |   Average Recall   | Average Precision  |
+---------+--------------------+--------------------+--------------------+--------------------+
|  Static | 0.8450483973516986 | 0.7988192307692307 | 0.9995641503856895 | 0.7320552537953664 |
| Dynamic |  0.85577835107624  | 0.8151692307692308 | 0.9991940093149109 | 0.748504056060965  |
+---------+--------------------+--------------------+--------------------+--------------------+
```

  The Limitations of the process are:

  - **Overfitting Risks:** Continuous learning might increase the risk of overfitting to the recent data, especially if the model is overly flexible. This could lead to reduced generalization performance.
  - **Dependency on Initial Model:** The retraining of the Dynamic model depends on the quality of the initial model. If the initial model is poorly fit, Continuous learning will not produce significant improvement.
  - **Retraining Time:** The retraining process of the model increases the time taken by the dynamic model after each retraining iteration, which might be problematic if it runs for a long time.



- **Knowledge Learned**

  In order to create a robust and adaptive model, we must start with a thorough understanding and cleaning of the data. Understanding the data is imperative as it provides insights on how to handle missing data, potential feature engineering, and feature selection. It is important to justify models used and understand limitations and assumptions of each model to choose the best model that will perform effectively on your data. The process of continuous model monitoring and evaluation is essential as it provides insights on when it’s necessary to retrain the model after falling below a preset threshold, allowing for continuous learning and deployment of a model that can generalize well.