Malicious Sessions Identification in SSL/TLS Encrypted Traffic

Introduction

Aim of this solution is to detect malicious sessions in SSL/TLS encrypted traffic.

Approach

In this project the following two approaches were tried:

- **Machine Learning models based on TLS handshake features**: In this approach I extract features from TLS handshake messages and apply machine learning classification techniques to classify between malicious and benign sessions. The idea behind this approach is to leverage the differences in the way TLS options are used between benign and malicious sessions for classification.
- **Deep Learning model based on payload as images**: In this approach I convert payload byte array into 2-dimensional image array and then feed the images to a CNN model to classify between malicious and benign sessions. *I did not have the compute to move forward with this method so I didn't explore this approach much. I have some lines of code in the end of the notebook to describe what I wanted to achieve but it doesn't work.*

Data Preprocessing

Scapy, which is a python based packet manipulation library, was used to extract TLS handshake fields and values. The packets from each session were concatenated to create a TLS stream and were fed to Scapy to extract information which was transformed into numerical and categorical features. Missing numerical fields were treated as absent and filled by zero.

Feature Selection

- **Machine Learning models based on TLS handshake features**

The following features were extracted from the TLS handshake messages. The TLS handshake messages used for feature extraction were Client Hello, Server Hello and Certificate:

1. client_cipher_len
2. client_ciphers
3. server_cipher

4. client_extension_len
5. client_extensions
6. server_extension_len
7. server_extensions
8. cert_len
9. subject_common_name_cnt

The above feature set is divided into two categories, which is described below:

1. **Numerical Features**
    a. 'client_cipher_len'
    b. 'client_extension_len'
    c. 'server_extension_len'
    d. 'Cert_len'
    e. 'subject_common_name_cnt'
2. **Categorical Features**
    a. 'Server_cipher',
    b. 'Client_ciphers',
    c. 'client_extensions',
    d. 'server_extensions

Out of the 9 features extracted above I decided to use all the numerical features and only one categorical feature i.e. server_cipher as part of my analysis. This was done for two reasons:

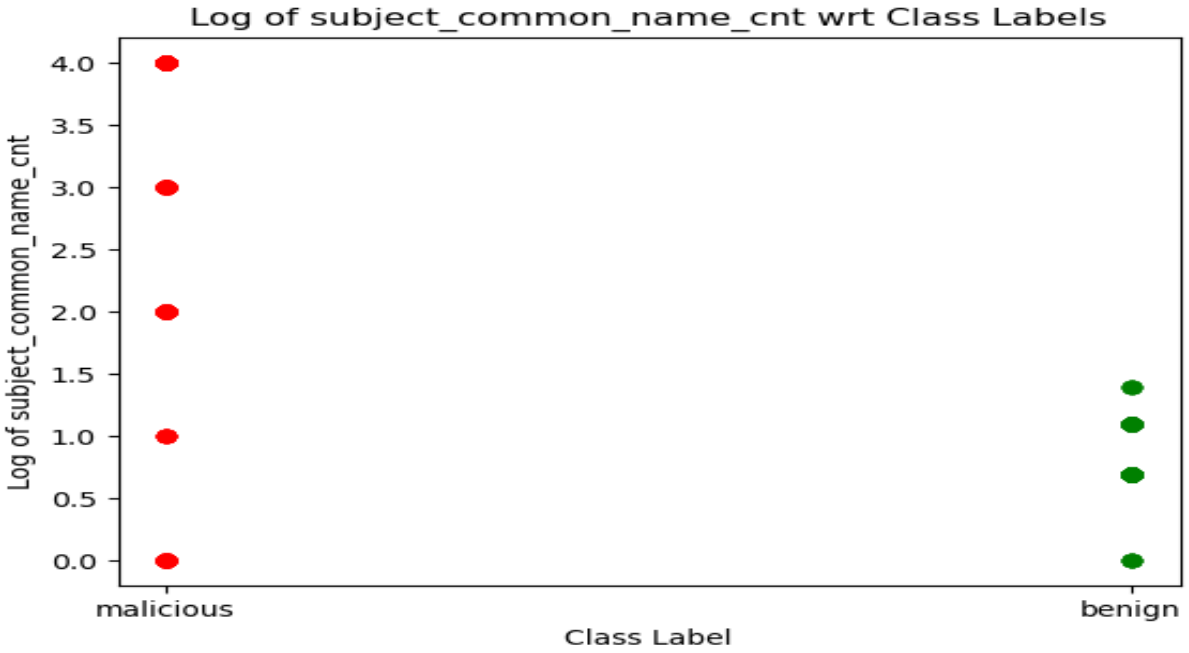
1. Other categorical features would have taken longer for me to encode and preprocess and to understand how they work
2. I would like to learn more about the TLS ciphers so that I can incorporate them with better understanding

Exploratory Data Analysis on the feature set

I have plotted the feature values against each class to see if there are any associations between the value of these features with the class labels, i.e. being malicious or benign.

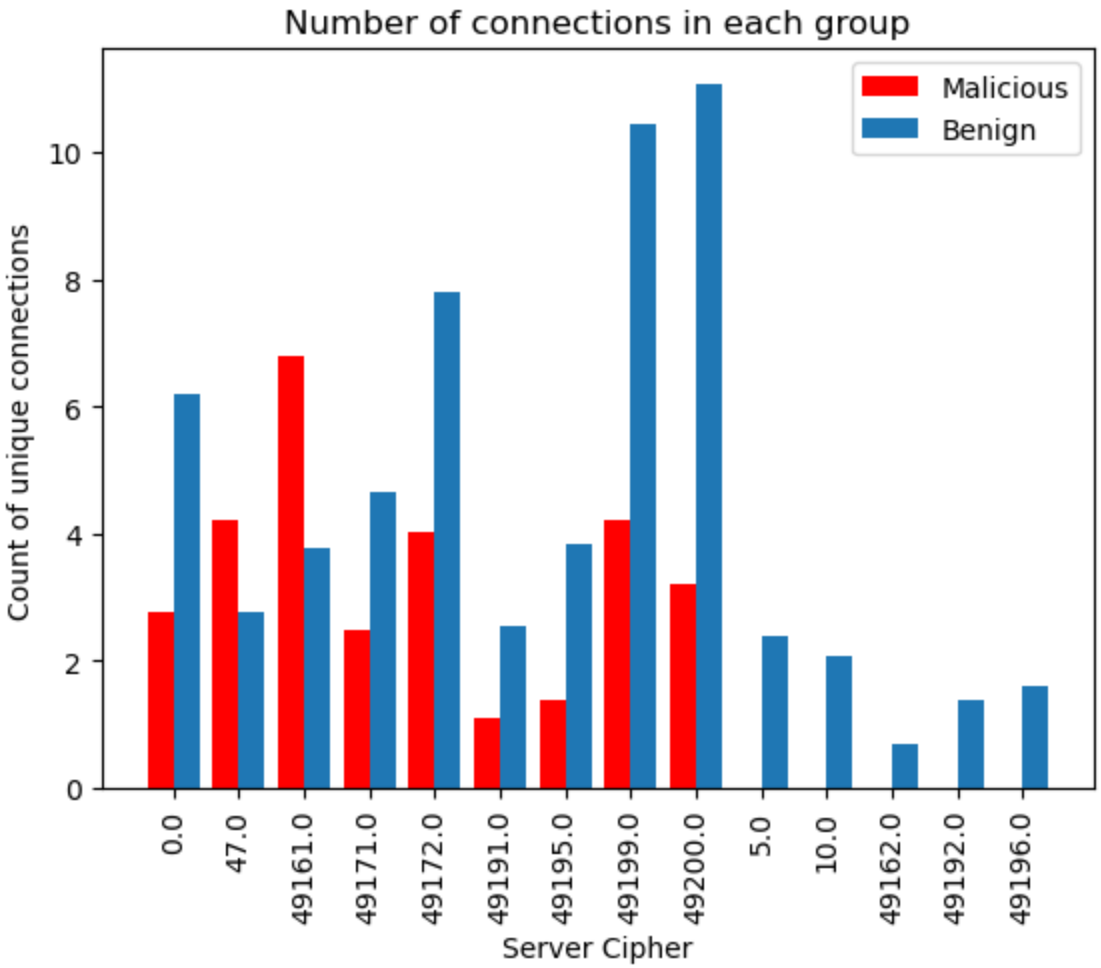
1. **'subject_common_name_cnt':** I have plotted the log values of this feature to see the minute differences that might be ignored in case of the absolute values. And we can see that the subject common name count associated with the benign class is very different from the subject common name count associated with the malicious class. *Hence this looks like a very important*

feature to be considered.



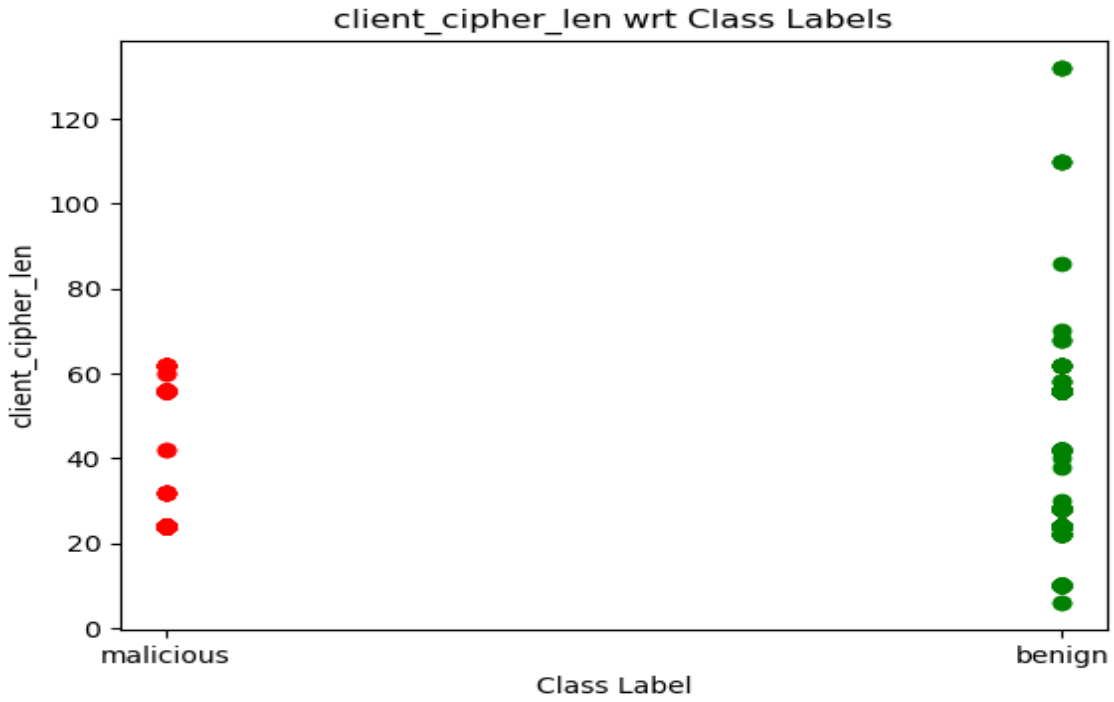
2. **Server_cipher:** This feature has some of the ciphers that give higher indication of being a malicious session than a benign connection like cipher 47 and 49161 but there are many that are associated with being benign only which I have excluded to include as features in my analysis. The plot below describes how Server cipher demonstrates characteristics of a session being malicious or benign. This feature doesn't provide a clear distinction but definitely gives some good information.

   This feature was one-hot encoded as it is a categorical variable.

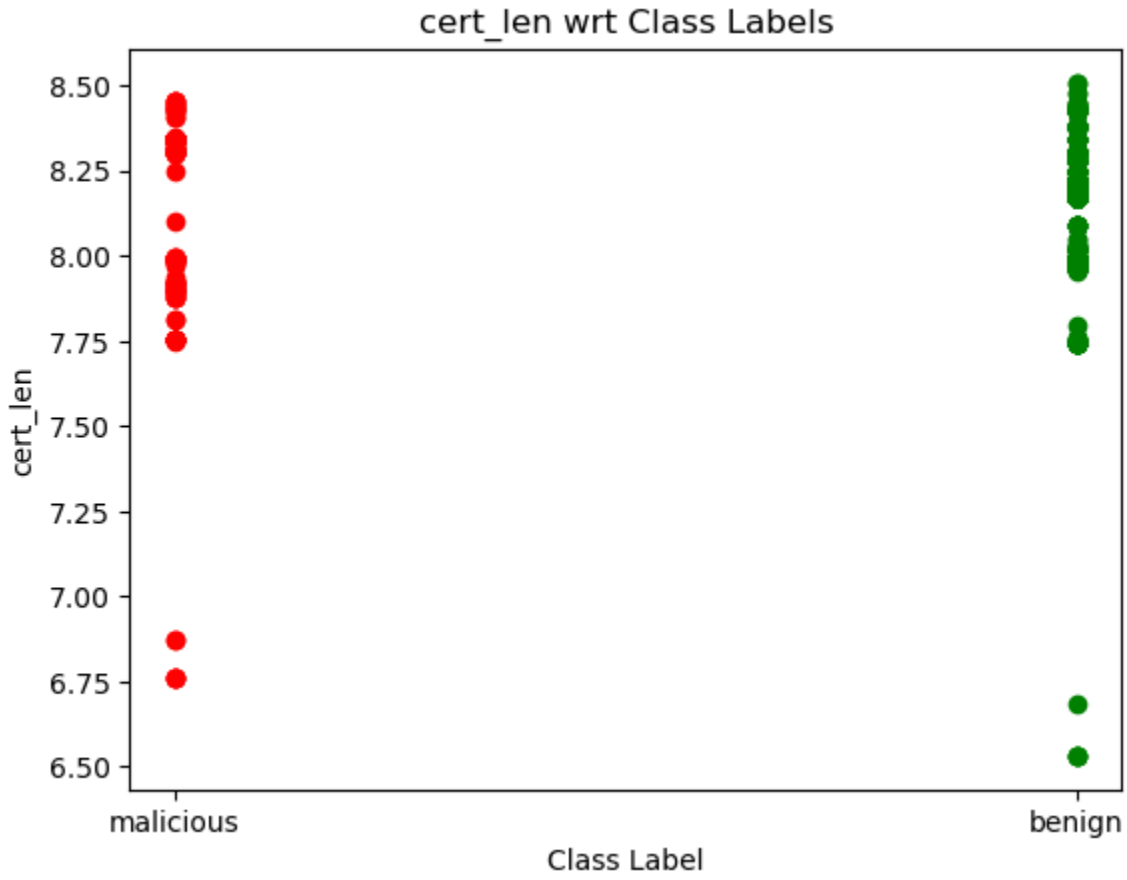


3. **Client_cipher_len:** It can be seen in the plot below that there is a range of client_cipher_len that is associated with malicious sessions and we can use

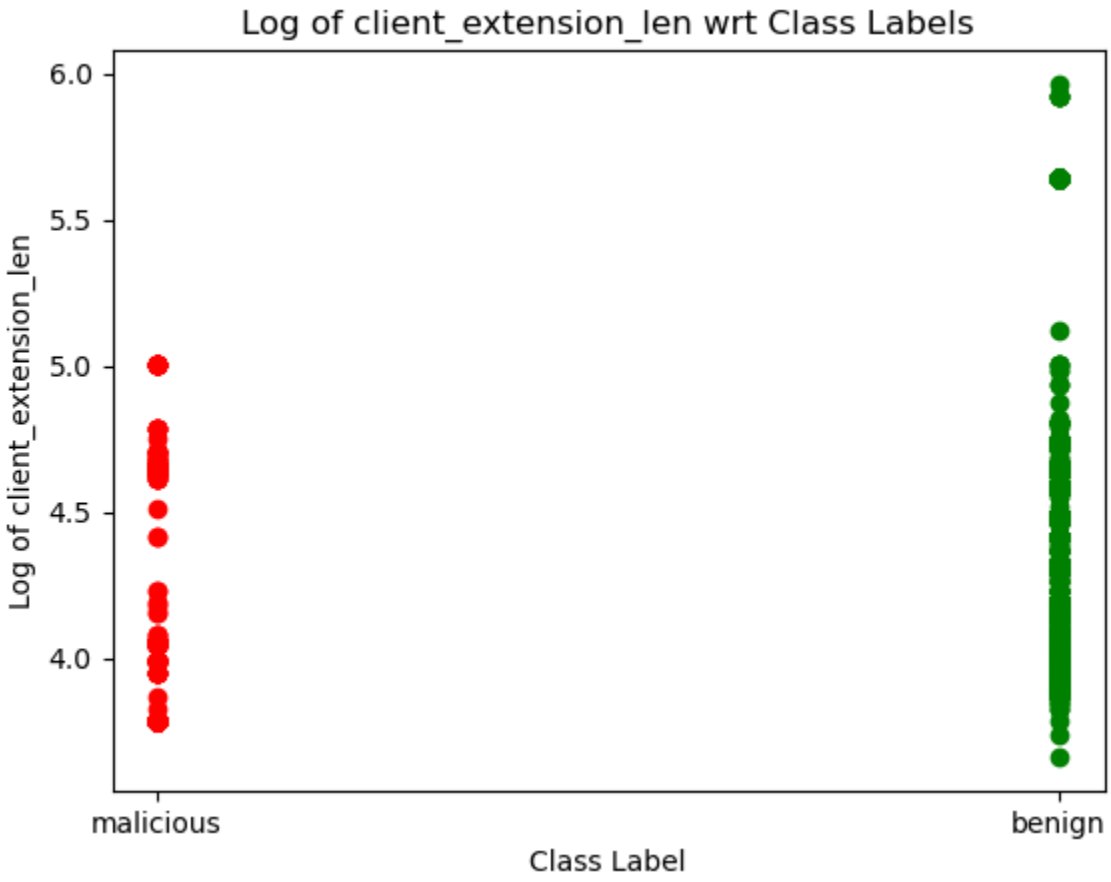
this info to feed into our model.



4. **Cert_len:** This feature doesn't seem to give us a lot of info but doesn't harm to include in the model. We can take it out if it doesn't help much as part of feature engineering.

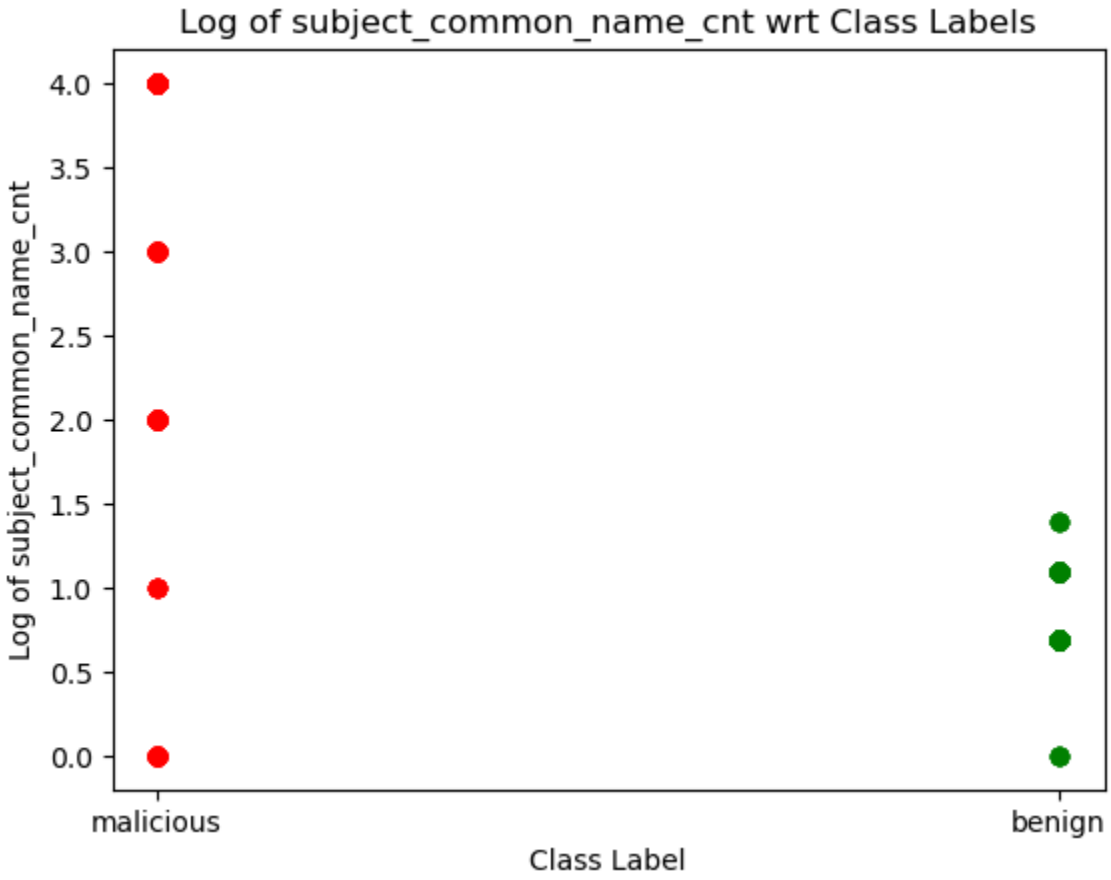


5. **Client_extension_len:** This feature also provides some range that can be associated with malicious traffic but it is not so clear.



6. **Server_extension_len:** Log of values were plotted to not miss the small differences in the values. This feature gives a good indication of differentiation

between malicious and benign sessions.



Sampling Technique

As part of this project I have used Stratified Sampling technique as there was a problem of class imbalance and I wanted to consider the same proportion of each class as in the actual data in my samples to help the models to generalize better.

Machine Learning Models

In this project I have tried to build the following machine learning models. This section describes the reasoning behind considering these models

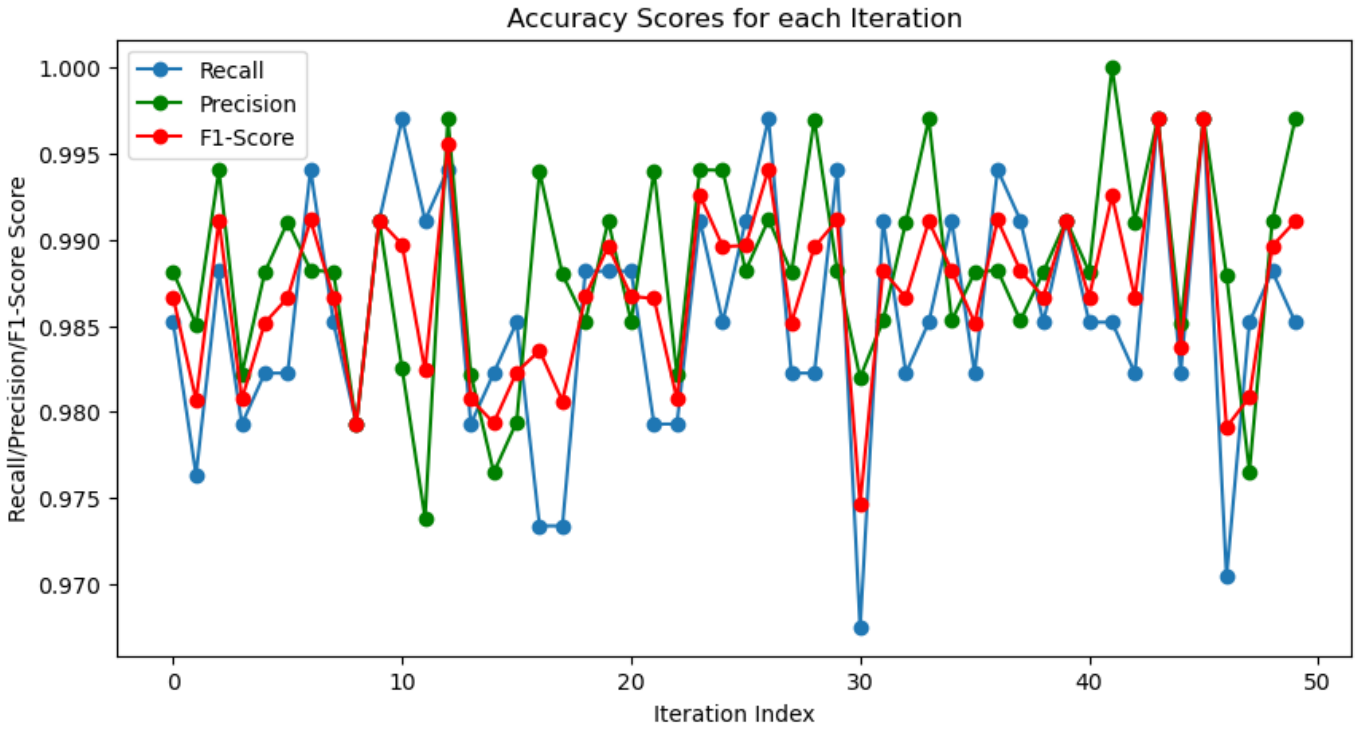
Random Forest Classifier

This is a tree based intuitive classifier which performs really well on large datasets and it automatically balances datasets when there is a problem of class imbalance. This model was a

good fit for this problem statement as the malicious class only constituted 1% of the total dataset.

Accuracy Score

- Recall - 98.6%
- Precision - 98.8%
- F1-Score - 98.7%

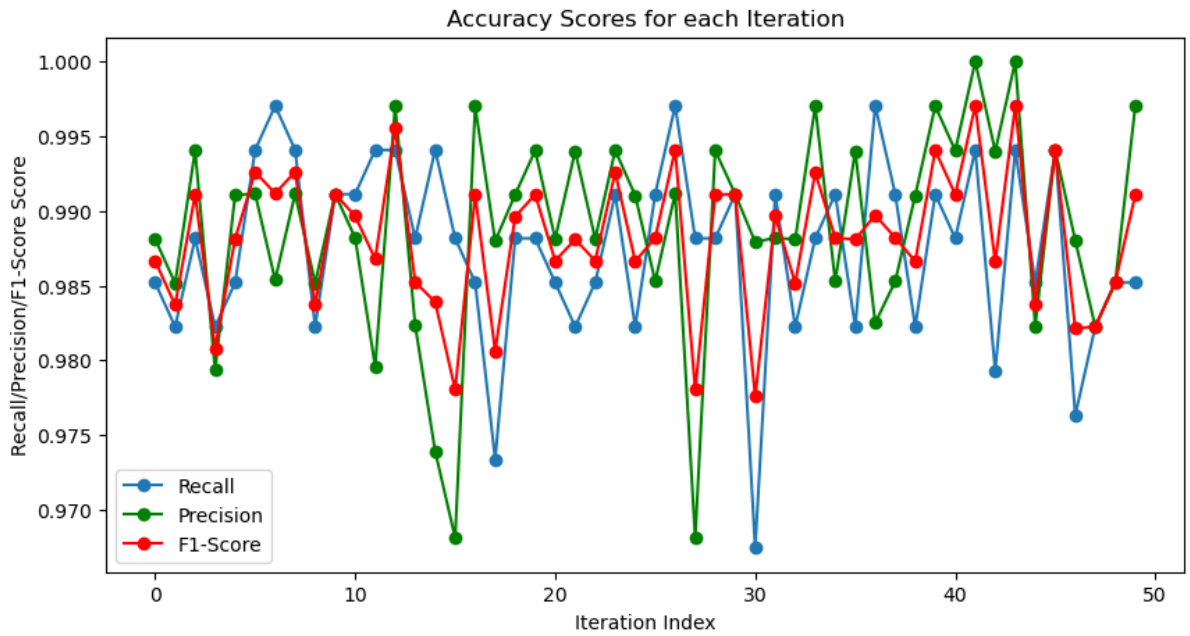


XGBoost Classifier

This model is only slightly better than Random Forest Classifier. The only reason behind using this classifier after Random Forest was that this classifier builds its subsequent trees based on the knowledge it gets from the previous trees. This makes it a more efficient model to be used and also in some cases can better avoid the problem of overfitting than Random Forest Classifier.

Accuracy Score

- Recall - 98.7%
- Precision - 98.9%
- F1-Score - 98.8%

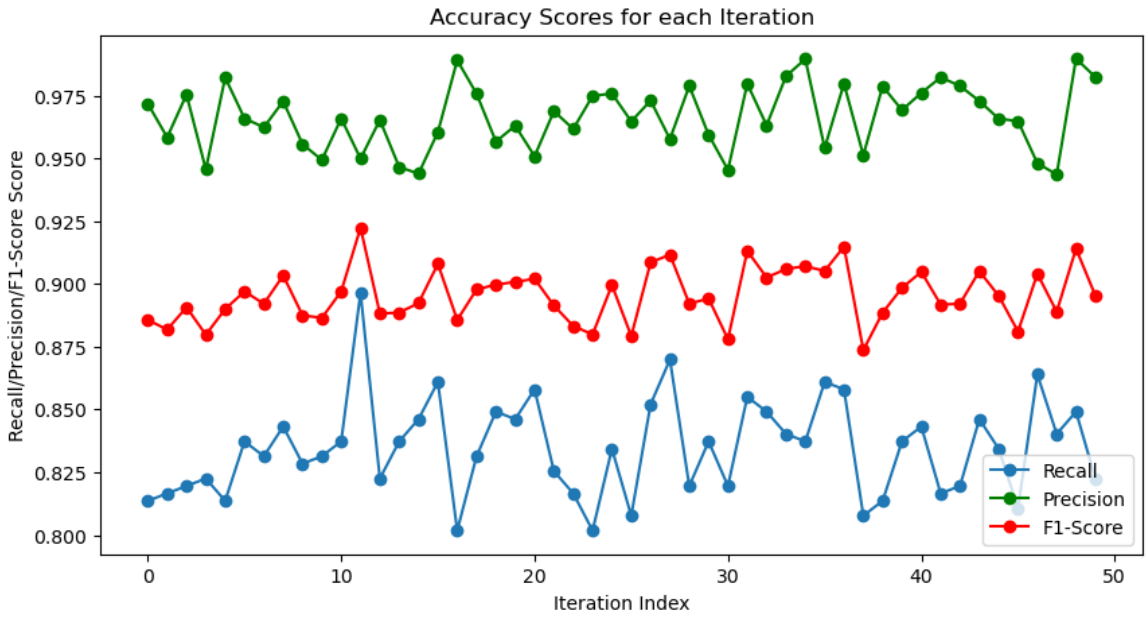


Logistic Regression Classifier

This classifier was not a great choice as it has an assumption of linearity between the logit of dependent variables and the independent variables which may not be true for some of the features in our case. It also gave the worst accuracy score out of all the three models that were explored in this project.

Accuracy Score

- Recall - 83.5%
- Precision - 96.6%
- F1-Score - 89.5%



Conclusion

XGBoost is well suited to solve this problem as it is efficient, intuitive as well as gives a good accuracy score of 98.8% F1-score. The accuracy score is determined in terms of Recall, Precision and F1-Score to overcome the class imbalance problem.

Future Work

1. Try the second approach of image classification using CNN
2. Try techniques like Oversampling, Undersampling or SMOTE to address the problem of class imbalance
3. Incorporate other categorical features in the analysis like 'Client_ciphers', 'client_extensions', 'server_extensions' with proper data preprocessing and encoding
4. Explore other TLS handshake messages to construct feature set