***D4.6 Final ENTA Solution for Use Case 1***

***Document: WP4-D4.6.0***

***Date: December 16, 2024***

Table of Contents

[List of Figures 3](#_Toc185405726)

[List of Tables 3](#_Toc185405727)

[Change Log 4](#_Toc185405728)

[**Summary** 5](#_Toc185405729)

[1. Introduction 6](#_Toc185405730)

[2. Summary of ENTA Solution before Enhancement 6](#_Toc185405731)

[3. Description of the Final ENTA Solutions for Use Case 1 6](#_Toc185405732)

[3.1. DL Solution for Use Case 1 6](#_Toc185405733)

[List of its Functional Tests 10](#_Toc185405734)

[3.1.1. List of Functional Test Results 11](#_Toc185405735)

[3.1.2. Model Performance - Single Head Architecture 11](#_Toc185405736)

[Comparison for single head models on the same data set 11](#_Toc185405737)

[Comparison for multi-head models on the same dataset 12](#_Toc185405738)

[Comparison for Multi-Head Models with Transfer Learning 13](#_Toc185405739)

[Key Observations: 13](#_Toc185405740)

[Conclusion: 13](#_Toc185405741)

[Overall Comparison 14](#_Toc185405742)

[3.1.3. Experimental dataset and feature extraction 14](#_Toc185405743)

[3.1.4. Use MLP/LSTM 14](#_Toc185405744)

[3.1.5. Model evaluation using train/test split on Solana2022a datasets 14](#_Toc185405745)

[3.1.6. Observation/Conclusion 14](#_Toc185405746)

[3.2. ML Solution for Use Case 1 Group Chat Classification 14](#_Toc185405747)

[3.2.1. A Description of the Dataset used 14](#_Toc185405748)

[3.2.2. A Summary of Feature Selection 14](#_Toc185405749)

[3.2.3. Selection, Training, and Testing of the Model 14](#_Toc185405750)

[3.2.4. Observation/Conclusion 14](#_Toc185405751)

[3.3. ML Classification for High-Speed Processing 14](#_Toc185405752)

[3.3.1. Flow Collection 14](#_Toc185405753)

[3.3.2. Data Preprocessing 14](#_Toc185405754)

[3.3.3. Feature Extraction 14](#_Toc185405755)

[3.3.4. Machine Learning Model 14](#_Toc185405756)

[4. Conclusion [To be updated] 14](#_Toc185405757)

[Appendix A 16](#_Toc185405758)

[Appendix B 16](#_Toc185405759)

[Appendix C 16](#_Toc185405760)

[Reference 17](#_Toc185405761)

[Acronym/Glossary 17](#_Toc185405762)

List of Figures

**No table of figures entries found.**

List of Tables

**No table of figures entries found.**

Change Log

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Submission date | Description of changes | Affected Sections |
| Initial Draft  WP4-D4.6.0 | 12-16-2024 | Initial draft | NA |

**Summary**

This document describes the final ENTA (Encrypted Network Traffic Analysis) solutions for the ENTA use case one. The solutions described in this document built upon the results reported in [D4.4](#D4_4) “Enhance ENTA solution for use case 1 – Beta” document. One of them uses ML-based model to detect group chat and the other one uses DL-based model to classify encrypted traffic type and category. As enabling in a timely manner in real-time use case 1 solutions and also other related solutions is highly desirable, this document also updates the preliminary solution for the machine-learning classification leveraging high-speed processing described in [D4.4](#D4_4).

1. Introduction

This document describes the final ENTA (Encrypted Network Traffic Analysis) solutions for the ENTA use case one. The solutions described in this document built upon the results reported in [D4.4](#D4_4) “Enhance ENTA solution for use case 1 – Beta” document. One of them uses ML-based model to detect group chat and the other one uses DL-based model to classify encrypted traffic type and category. As enabling in a timely manner in real-time use case 1 solutions and also other related solutions is highly desirable, this document also updates the preliminary solution for the machine-learning classification leveraging high-speed processing described in [D4.4](#D4_4).

1. Summary of ENTA Solution before Enhancement

The use case 1 solution described herein handles activities transpiring during the usage of network traffic applications. This enhancement of detecting group chat is updated in Section 3.2. Moreover, a use case 1 solution using DL-based model is updated in Section 3.1. This document also updates the preliminary solution for machine-learning classification for high-speed processing described in [D4.4](#D4_4) in Section 3.3. This latter solution enables in a timely manner in real-time use case 1 solutions which is also highly desirable for other encrypted network traffic analysis use cases.

1. Description of the Final ENTA Solutions for Use Case 1
   1. DL Solution for Use Case 1

As outlined in Section 3.2.2 of the D4.4 document, Multi-Layer Perceptron (MLP), Long Short-Term Memory (LSTM), and Convolutional Neural Network (CNN) models were implemented to classify encrypted traffic application types. These deep learning-based solutions are aimed at identifying traffic patterns in encrypted data, making it possible to distinguish between different application types, even when the data is encrypted.

**Dataset Collection**

To train and evaluate the models, we used Encrypted Mobile Instant Messaging Traffic Dataset, which was collected by NIMS Lab, Dalhousie University. The encrypted traffic was collected using six widely-used Instant Messaging Applications (IMAs) installed on an Android device. Specifically, the dataset focused on **Text Chat traffic**, capturing data from the following applications:

1. **Microsoft Teams**
2. **Discord**
3. **Facebook Messenger**
4. **Signal**
5. **Telegram**
6. **Whatsapp**

In addition to traffic from IMAs, other forms of encrypted mobile traffic were also captured to provide a broader context for classification. These non-IMA traffic types were collected from the following sources:

* **Mail Traffic:** Data was collected from the **Gmail** application to represent encrypted email traffic.
* **Streaming Traffic:** Traffic generated from video streams was captured using the **YouTube** application.
* **Web Browsing Traffic:** Data was gathered from general web browsing across various websites.
* **Background Traffic:** This category included traffic generated by background processes not tied to any of the above applications or specific user interactions.

The captured data was stored in **PCAP** (Packet Capture) files, which is a standard format for storing network traffic data. To facilitate analysis and model training, various features were extracted from the **bidirectional flows** using the **NFStream** feature extractor. This tool is specifically designed for efficient flow-based network traffic analysis.

Two subsets of features (Appendix – A) were selected from NFStream's default feature set for this analysis:

1. **Statistical Features:** These features provide aggregate statistics about the flow, such as packet sizes, inter-arrival times, and the total number of packets. These metrics offer insight into the overall characteristics of the traffic.
2. **Sequence Features:** These features capture the temporal order of packets and related information within the flow, enabling the models to leverage the sequence or timing of packets to differentiate between traffic types and applications.

These features were essential for training the deep learning models (MLP, LSTM, and CNN), allowing them to learn the patterns of encrypted traffic and accurately classify both application types and traffic classes.

**Data Preparation and Preprocessing**

Before the models were trained, the raw traffic data underwent a preprocessing step to ensure the quality and relevance of the data:

* Filtering of DNS and other networking flows**:** DNS (Domain Name System) flows and other non-relevant networking traffic were removed, as these did not contribute to the classification of encrypted application traffic.
* Filtering based on total packet count: Any bi-directional flow with a total packet count of 2 or fewer packets was filtered out. These short flows were not considered for meaningful analysis, as they do not provide enough information for classification.
* **Padding**: In the case of packet sequence features used for the LSTM and CNN models, a sequence length of 100 was employed. If a flow contained fewer than 100 packets, the remaining sequence positions were padded with the value '0' to ensure that each sequence had a uniform length of 100.

**Data Labeling**

Furthermore, the bi-directional flows from each PCAP file were labeled based on the type of application from which the data was captured. As a result, the dataset contains traffic from seven application types. Table 1 shows the bi-directional flow count for train and test datasets for each of the application.

Table 1 Train/Test Data Flow Count

|  |  |  |
| --- | --- | --- |
| Application | Train Dataset (# of flows) | Test Dataset (# of flows) |
| Discord | 3513 | 894 |
| Facebook Messenger | 3499 | 859 |
| Microsoft Teams | 14753 | 3733 |
| Others | 7901 | 1958 |
| Signal | 4027 | 1011 |
| Telegram | 4827 | 1209 |
| Whatsapp | 2550 | 621 |

The "Others" label was assigned to traffic collected from non-IMA (Instant Messaging Application) sources, such as streaming from YouTube, web browsing, and background traffic. This labeling was critical for the classification task, enabling the models to distinguish between various IMAs and non-IMA traffic, contributing to a more comprehensive analysis of encrypted traffic patterns.

**Model Training and Evaluation**

The collected dataset underwent descriptive and statistical analysis to prepare the data for model training. This involved preprocessing steps such as traffic segmentation, feature extraction, and encoding. Each deep learning model—MLP, LSTM, and CNN—was trained on this dataset to learn the distinctive patterns of traffic associated with each application and traffic type.

* The MLP model was designed to capture general patterns and relationships within the encrypted traffic data through fully connected layers. These features, described in Appendix A, capture high-level traffic characteristics, providing the MLP model with general insights into traffic behavior. These features played a critical role in helping the MLP model identify patterns in the encrypted traffic.
* The LSTM and CNN models leveraged their ability to retain sequential information, making them particularly effective for time-dependent traffic analysis.
* Additionally, CNN models were explored using statistical features. By processing these high-level traffic characteristics, the CNN model was able to identify spatial patterns within the statistical data, further enhancing the analysis of encrypted traffic.

The models were then evaluated based on their ability to classify both the IMA-specific traffic (e.g., from Microsoft Teams, Discord, and Telegram) and non-IMA traffic (e.g., Gmail, YouTube, web browsing, and background processes). The results demonstrated the performance of the models in distinguishing between encrypted traffic from different applications, even without access to the actual content.

In the following, we list the functional tests to be considered, describe how the tests to be completed, and record their results obtained. This subsection ends with some conclusion on the functional test performed.

**Test Case Design**

1. **Single-Head Model Testing:**

Each single-head model was trained and tested independently to establish baseline performance:

* + **MLP:** Used statistical features to capture general traffic patterns and high-level characteristics.
  + **LSTM:** Processed packet sequence features to leverage its ability to model sequential and time-dependent patterns.
  + **CNN (Packet Sequence):** Focused on packet sequence features to identify spatial relationships in sequential data.
  + **CNN (Statistical):** Explored statistical features to identify spatial patterns within high-level traffic characteristics.

1. **Multi-Head Model Testing:**

Multi-head architectures were designed to combine the strengths of different models, processing diverse feature types through parallel branches. The combinations tested included:

* + **MLP and LSTM:** Integrated statistical features (MLP) with sequential patterns (LSTM) to capture both general and temporal insights into the traffic data.
  + **MLP and CNN (Packet Sequence):** Combined statistical features processed by MLP with spatial relationships in packet sequence data captured by CNN.
  + **LSTM and CNN (Statistical):** Paired LSTM’s sequential analysis capability with CNN’s ability to extract spatial patterns from statistical features.

1. **Enhanced Multi-Head Testing with Transfer Learning:**

* **Pre-Trained Models from Single-Head Tests:** Pre-trained single-head models from item 1 were used as starting points for the respective branches in the multi-head architectures. For example:
  + Pre-trained MLP on statistical features was used in the MLP branch.
  + Pre-trained LSTM on packet sequence data was used in the LSTM branch.
  + Pre-trained CNNs (both statistical and packet sequence-based) were used in the respective CNN branches.
* **Transfer Learning in Multi-Head Combinations:** Transfer learning was applied to the same multi-head combinations outlined in item 2, leveraging the pre-trained models for each branch. This ensured that each branch started with a strong, task-specific feature representation.
* **Layer-Freezing Strategies:** Some layers in the pre-trained branches were frozen to retain previously learned feature representations, while the remaining layers were fine-tuned for the combined task of multi-head classification.
* **Benefits of Transfer Learning:** The use of pre-trained models enhanced convergence speed, reduced the need for extensive training, and improved overall classification performance, especially for smaller datasets.

### List of its Functional Tests

To evaluate the effectiveness of the new methodologies—multi-head architectures and transfer learning—additional functional test cases were introduced alongside the baseline tests from the previous report.

**Train and Test on the Same Dataset:**

* In this test, the collected dataset was split into a train-test split of 80-20%, where 80% of the data was used for training and 20% for testing. This setup provided a baseline for evaluating model performance.
* The training process varied depending on the model architecture and test case type:
* **Single-Head Models:**
  + **MLP:** Statistical features extracted from NFStream were used to train the MLP model. These features captured high-level traffic characteristics and provided insights into traffic behaviour.
  + **LSTM:** Packet sequence features, containing time-dependent patterns, were used to train the LSTM model, leveraging its ability to handle sequential data.
  + **CNN (Packet Sequence):** The CNN model trained on packet sequence features identified spatial relationships in the sequential data.
  + **CNN (Statistical):** A CNN model trained on statistical features was introduced to analyse spatial patterns in high-level traffic characteristics.
* **Multi-Head Models:** Multi-head architectures combined the strengths of different models to process diverse feature types through parallel branches. Tested combinations included:
  + **MLP and LSTM:** Statistical features were processed by the MLP branch, while sequential patterns were analysed by the LSTM branch.
  + **MLP and CNN (Packet Sequence):** Statistical features were handled by the MLP branch, while packet sequence features were analysed by the CNN branch.
  + **LSTM and CNN (Statistical):** Sequential patterns were handled by the LSTM branch, while statistical features were processed by the CNN branch to extract spatial patterns.
* **Enhanced Multi-Head Models with Transfer Learning:**
  + Pre-trained single-head models (trained in the earlier tests) were used as starting points for respective branches in the multi-head architectures. For example:
    - Pre-trained MLP models on statistical features were integrated into the multi-head models.
    - Pre-trained LSTM and CNN models were incorporated to analyse packet sequence or statistical features as appropriate.
  + Transfer learning techniques included freezing specific layers in pre-trained branches to retain previously learned features, while fine-tuning the remaining layers for task-specific optimization.

**Train and Test on Different Datasets:**

To further assess the generalizability and robustness of the new methodologies, additional tests were conducted using separate datasets for training and testing:

* + **Dataset Splitting:** A subset of the collected dataset was designated for training, while an entirely different subset was used for testing. This ensured that the testing dataset was unseen during training.
  + **Objective:** These tests evaluated the ability of single-head and multi-head models, especially those using transfer learning, to generalize beyond the training data and perform well on new, unseen traffic samples.
  + **Objective on Transfer Learning:** The transfer learning-based multi-head models were expected to exhibit enhanced performance due to their ability to leverage pre-trained knowledge for better adaptation to unseen datasets.
    1. List of Functional Test Results

This section summarizes the results of the functional tests performed to classify encrypted traffic data using the three deep learning models: **MLP**, **LSTM**, and **CNN.** The models were tested using the dataset described in previous sections, and their performance was evaluated based on several key metrics derived from the confusion matrix:

* **Recall**: Indicates the ability of the model to correctly identify instances of each class.
* **Precision**: Provides the proportion of true positive predictions among all positive predictions made by the model.
* **Weighted Precision**: It is average precision weighted by the number of instances for each class.
* **F1 Score**: The harmonic means of precision and recall, providing a single metric that balances both concerns.
  + 1. Model Performance - Single Head Architecture

### Comparison for single head models on the same data set

When evaluating the models (MLP, LSTM, and CNN (Packet Sequence Features) on a sample from the same dataset, the following observations are noted regarding their F1 Scores:

|  |  |  |  |
| --- | --- | --- | --- |
| Class | F1 Score (%)  MLP | F1 Score (%)  LSTM | F1 Score (%)  CNN |
| **Discord** | 86.19 | 92.15 | 92.56 |
| **Others** | 78.99 | 96.79 | 98.5 |
| **Telegram** | 66.16 | 89.73 | 92.69 |
| **Microsoft Teams** | 72.73 | 88.15 | 92.06 |
| **WhatsApp** | 92.65 | 98.93 | 99.57 |
| **Facebook Messenger** | 62.24 | 94.81 | 95.49 |
| **Signal** | 67.14 | 85.52 | 87.18 |

**Key Observations:**

* **CNN** generally outperformed both **MLP** and **LSTM**, achieving the highest F1 Scores across all classes, particularly for **WhatsApp (99.57%)** and **Others (98.50%)**.
* **LSTM** showed strong performance, especially for **Others (96.79%)** and **Facebook Messenger (94.81%)**, indicating effective handling of sequential data.
* **MLP** had the lowest F1 Scores for most classes, showing that while it captures general patterns, it is less effective compared to the other models for classes like **Telegram (66.16%)** and **Facebook Messenger (62.24%)**.
* **WhatsApp** was the best-classified class across all models, with each model achieving a strong F1 Score.
* The **CNN** model showed a clear advantage in distinguishing between encrypted traffic types, particularly in complex classes.

This comparison illustrates the benefits of using LSTM and CNN architectures for capturing sequential and spatial features in packet sequence data, while MLP is more suited for general traffic patterns.

### Comparison for multi-head models on the same dataset

The performance of multi-head models **MLP-LSTM** and **MLP-CNN (pkt seq) and LSTM-CNN (Statistical)** when evaluated on the same dataset is summarized below:

|  |  |  |  |
| --- | --- | --- | --- |
| Class | F1 Score (%)  MLP-LSTM | F1 Score (%)  MLP-CNN (Pkt Seq) | F1 Score (%)  LSTM-CNN (Stat) |
| **Discord** | 91.86 | 92.35 | TBD |
| **Others** | 96.93 | 97.91 | TBD |
| **Telegram** | 90.05 | 92.92 | TBD |
| **Microsoft Teams** | 90.06 | 91.07 | TBD |
| **WhatsApp** | 98.94 | 99.28 | TBD |
| **Facebook Messenger** | 93.67 | 96.21 | TBD |
| **Signal** | 86.07 | 87.6 | TBD |

**Key Observations:**

* **MLP-CNN** outperformed **MLP-LSTM** across all traffic classes, highlighting the superior performance of the CNN module when combined with MLP.
* The **WhatsApp** class achieved the best classification in both models, with **MLP-CNN** achieving an F1 Score of **99.28%**.
* **Others** and **Telegram** classes showed notable improvement in the **MLP-CNN** model, achieving F1 Scores of **97.91%** and **92.92%**, respectively.
* **Facebook Messenger** saw a significant gain with **MLP-CNN (96.21%)**, compared to **MLP-LSTM (93.67%)**.
* **Signal** had the lowest F1 Scores in both models, but **MLP-CNN (87.60%)** showed a slight improvement over **MLP-LSTM (86.07%)**.

### Comparison for Multi-Head Models with Transfer Learning

The performance of multi-head models **MLP-CNN** and **MLP-LSTM** with **transfer learning** is summarized below:

|  |  |  |  |
| --- | --- | --- | --- |
| Class | F1 Score (%)  MLP-LSTM | F1 Score (%)  MLP-CNN (Pkt Seq) | F1 Score (%)  LSTM-CNN (Stat) |
| **Discord** | 92.18 | 92.1 | TBD |
| **Others** | 96.87 | 96.99 | TBD |
| **Telegram** | 89.9 | 90.24 | TBD |
| **Microsoft Teams** | 89.46 | 88.97 | TBD |
| **WhatsApp** | 98.86 | 98.91 | TBD |
| **Facebook Messenger** | 95.5 | 95.42 | TBD |
| **Signal** | 86.27 | 85.64 | TBD |

### Key Observations:

* **Discord**: Both models performed nearly identically, with **MLP-LSTM** (92.18%) slightly outperforming **MLP-CNN** (92.10%).
* **Others**: The **MLP-CNN** model achieved a marginally higher score (96.99%) compared to **MLP-LSTM** (96.87%).
* **Telegram**: The **MLP-CNN** model slightly outperformed **MLP-LSTM** (90.24% vs. 89.90%).
* **Microsoft Teams**: **MLP-LSTM** had a slight edge over MLP-CNN (89.46% vs. 88.97%).
* **WhatsApp**: Both models performed exceptionally well, with almost identical results (**98.86%** for MLP-LSTM and **98.91%** for MLP-CNN).
* **Facebook Messenger**: **MLP-LSTM** achieved a slightly higher score (95.50%) compared to MLP-CNN (95.42%).
* **Signal**: **MLP-LSTM** (86.27%) performed better than MLP-CNN (85.64%).

### Conclusion:

In this comparison, both **MLP-LSTM** and **MLP-CNN** multi-head models demonstrated **high accuracy** and consistency across all traffic classes when enhanced with transfer learning. Key points include:

* **MLP-LSTM** showed slightly better results for **Discord**, **Microsoft Teams**, **Facebook Messenger**, and **Signal**.
* **MLP-CNN** performed marginally better for **Others** and **Telegram**, and achieved the best result for **WhatsApp**.

The results highlight that both architectures benefit from **transfer learning**, achieving robust classification performance with minor variations across classes.

### Overall Comparison

At the end of testing, the following insights will be derived:

* **Impact of Multi-Head Models**: Did combining architectures (e.g., MLP-LSTM, MLP-CNN) outperform single-head models?
* **Effectiveness of Transfer Learning**: What additional gains were observed when using pre-trained models?
* **Generalization Capability**: Which models performed best when tested on different datasets?
* **Class-Specific Performance**: Which classes (traffic types) consistently achieved high scores, and which remained challenging?
* **Overall Best Model**: Identify the model architecture and strategy (single-head, multi-head, transfer learning) that provided the best overall F1 score.
  + 1. Experimental dataset and feature extraction
    2. Use MLP/LSTM
    3. Model evaluation using train/test split on Solana2022a datasets
    4. Observation/Conclusion
  1. ML Solution for Use Case 1 Group Chat Classification
     1. A Description of the Dataset used
     2. A Summary of Feature Selection
     3. Selection, Training, and Testing of the Model
     4. Observation/Conclusion
  2. ML Classification for High-Speed Processing
     1. Flow Collection
     2. Data Preprocessing
     3. Feature Extraction
     4. Machine Learning Model

1. Conclusion [To be updated]

This document presents two enhancements made to the current ENTA platform software and two enhanced solutions for two use case 1 problems.

The ENTA platform enhancements consist of supporting the definition of new user-defined feature extractor and DL pipeline.

The two use case 1 solutions consist of detecting group chat using ML-based model and classifying encrypted network traffic (according to traffic type and category) using DL-based model.

The above enhancements show the extensibility of the ENTA platform software and its applicability to diverse encrypted traffic analytic problems.

To enable in real-time use case 1 solutions in a timely manner, a preliminary high-speed processing solution for machine-learning classification is proposed.

## Appendix A

## Appendix B

## Appendix C

Reference

|  |  |
| --- | --- |
| [D4.4] | D4.4 “Enhance ENTA solution for use case 1 – Beta” Deliverable Document  Available at: <https://itea4.org/community/project/workpackage/deliverable/document/download/395/D4.4%20Enhanced%20ENTA%20Solution%20for%20Use%20Case%201%20-Beta-v2.pdf> |

Acronym/Glossary

|  |  |
| --- | --- |
| Acronym | Meaning |
| ACK | Acknowledge, used to confirm to the other end that a SYN has been received in a TCP connection |
| Benford’s Law | A phenomenological law also called the first digit law, first digit phenomenon, or leading digit phenomenon. (<https://mathworld.wolfram.com/BenfordsLaw.html>) |
| CNN | Convolutional Neural Network |
| CSV | Comma-Seperated Value |
| CWR | Congestion Window Reduced, used to indicate the congestion window has been reduced. |
| DL | Deep-Learning |
| ECE | Used to indicate congestion experienced. |
| ENTA | Encrypted Network Traffic Analysis |
| Feast | <https://docs.feast.dev/reference/data-sources/overview> |
| FIN | Indicates the end of data transmission when finishing a TCP connection |
| FN | False Negative |
| FP | False Positive |
| GBM | Gradient-Boosting Machine |
| HSPP | High-Speed Packet Processing |
| IMA | Instant Messaging Application |
| JSON | JavaScript Object Notation  <https://www.json.org/json-en.html> |
| KFP | Kubeflow Pipelines |
| LSTM | Long-Short Term Memory |
| MinIO | MinIO is a high-performance, S3 compatible object store. It is built for large scale AI/ML, data lake and database workloads. It is software-defined and runs on any cloud or on-premises infrastructure. (<https://min.io/>) |
| ML | Machine-Learning |
| MLP | Multilayer Perceptron |
| PSH | Push, a segmentation bit |
| ReLU | Rectified Linear Unit |
| RF | Random Forest |
| RST | Reset. used to end a TCP sesion. |
| SDK | Software Development Kit |
| SKLearn | <https://scikit-learn.org/stable/> |
| SVM | Support Vector Machine |
| SYN | Synchronize, used to initiate and establish a TCP connection. |
| TN | True Negative |
| TP | True Positive |
| UDFE | User Defined Feature Extractor |
| URG | Urgent, used to signal “urgent” data should be prioritized over non-urgent data. |
| XGBoost | <https://xgboost.readthedocs.io/en/stable/> |