VPN traffic classification

Made by Denys Tereshchenko
Following course of lectures
“Introduction to machine learning”

Problem statement

Problem statement

TLS/SSL

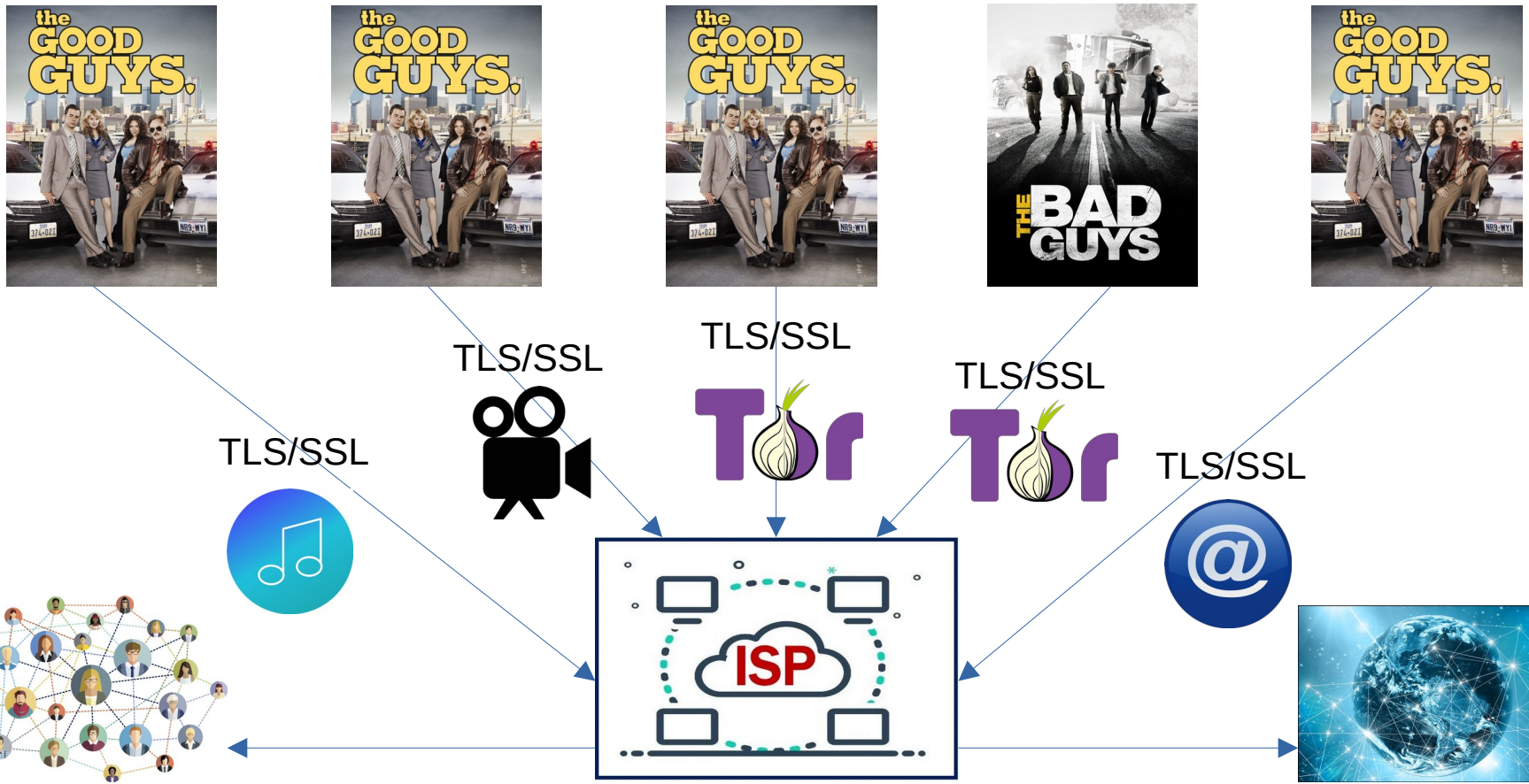
TLS/SSL

TLS/SSL

TLS/SSL

TLS/SSL

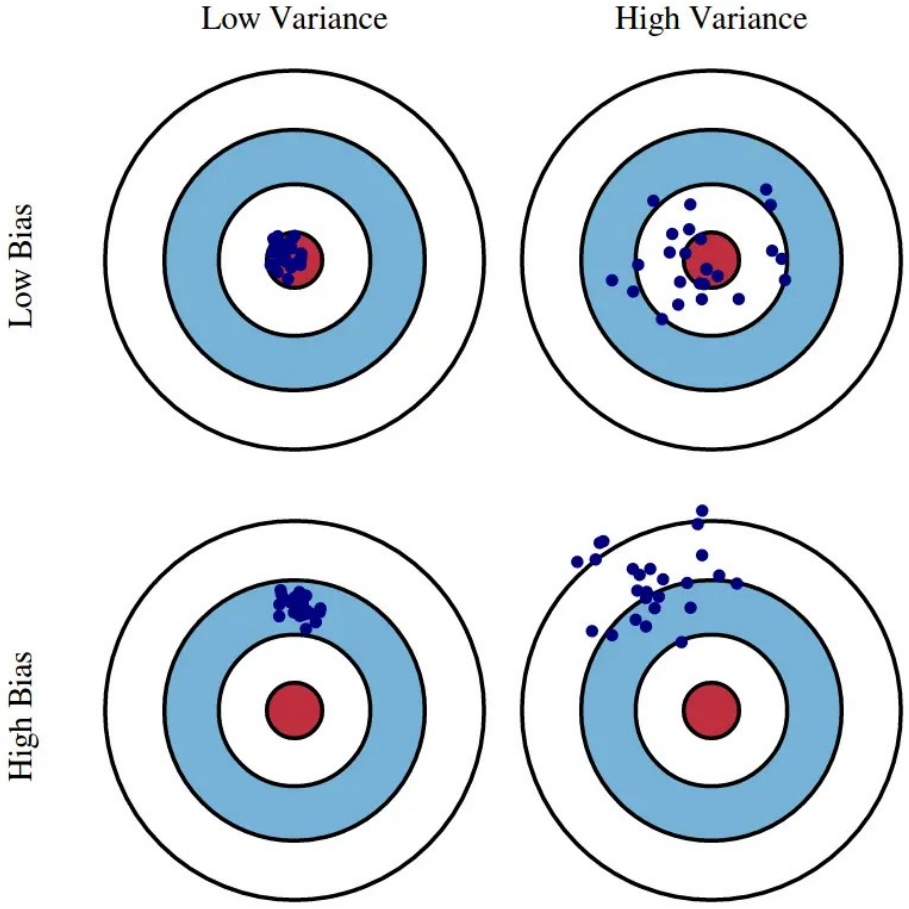
Problem statement

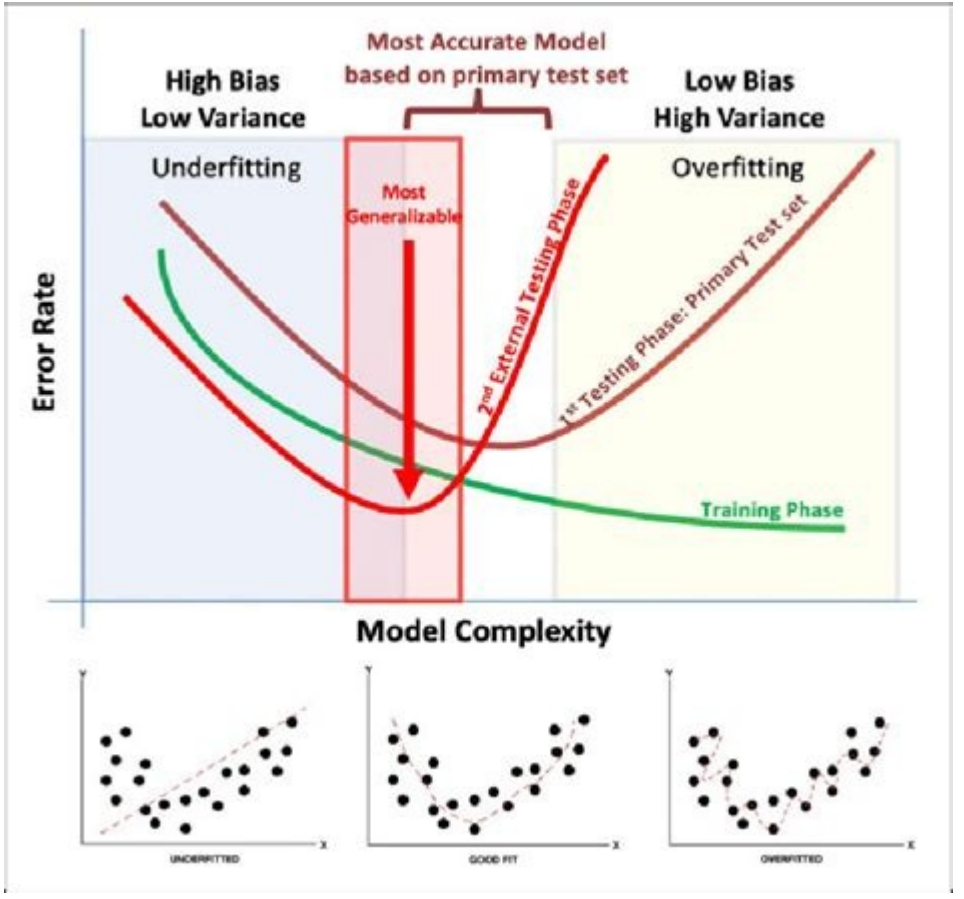


Existing approaches

Granularity	Algorithm	Input	Dataset	Early Stage Detection
Application	SAE	Raw traffic	Private	N/A
Application	CNN	Raw traffic	USTC-TFC2016	784 bytes
Service group	CNN	Raw traffic	ISCX	784 bytes
Application	CNN, LSTM	Packet level features	RedIRIS	10 packets
Application/Service Group	CNN/SAE	Raw traffic	ISCX	N/A
Application	CNN	Raw traffic	CTU-13/ISCX	1024 bytes
Application	CNN/MLP/SAE	Raw traffic	ISCX	1480 bytes
Application	CNN, SAE, LSTM	Raw/Packet level feature	Private	784 bytes
Service Group	CNN	Flow statistical features	Moore	No
Application	Random Forest/CNN	Flow/Packet features	Private	No
Protocol	CNN	Raw traffic	Private	10 Packets
Application	CNN, GRU	Raw traffic, time series features	Private	576 bytes/12 packets
Application	CNN, RNN	Raw traffic, time series features	Private	576 bytes/12 packets
Application	NIN, MLP	Raw traffic	ISCX	1480 bytes

Types of errors





Types of errors

$$\mathrm{E}\left[\left(y - \hat{f}\left(x\right)\right)^2\right] = \mathrm{Bias}\left[\hat{f}\left(x\right)\right]^2 + \mathrm{Var}\left[\hat{f}\left(x\right)\right] + \sigma^2$$

Where:

$$\mathrm{Bias}\left[\hat{f}\left(x\right)\right] = \mathrm{E}\left[\hat{f}\left(x\right) - f(x)\right]$$

and

$$\mathrm{Var}\left[\hat{f}\left(x\right)\right] = \mathrm{E}[\hat{f}\left(x\right)^2] - \mathrm{E}[\hat{f}\left(x\right)]^2$$

Criticism

Analysis of the ISCX VPN-nonVPN Dataset 2016 for Encrypted Network Traffic Classification

Felipe Peter
Tsinghua University
felipe.peter@tum.de

https://github.com/Mr-Pepe/iscx-analysis/blob/master/report.pdf

Criticism

Application	Number of flows	Number of packets	Unique flows	Unambiguous packets
AIM chat	458	5601	0.13	0.37
Email	3057	51098	0.33	0.55
Facebook	22253	2601864	0.17	0.88
FTPS	1291	6038433	0.9	1.0
Gmail	250	9931	0.74	0.94
Hangouts	21613	4689219	0.16	0.94
ICQ	495	8096	0.15	0.5
Netflix	286	732836	0.86	0.23
SCP	5769	415955	0.26	0.93
SFTP	196	855456	0.72	1.0
Skype	26473	4466639	0.28	0.92
Spotify	299	97470	0.93	0.25
Torrent	2097	341433	0.9	0.95
Tor	42	210605	0.86	1.0
VoipBuster	3823	1563872	0.36	0.99
Vimeo	551	366694	0.89	0.86
Youtube	803	268310	0.95	0.69
Total/ Average	89756	22723512	0.27	0.92

Table 3. Results of the dataset analysis based on flows consisting of the triplet source port, destination port, and used protocol. While the fraction of unique flows is relatively low, 92% of the packets can be uniquely associated with one application. This analysis incorporates both VPN and non-VPN traffic.

Criticism

5. Conclusion

This work has shown that an approach using the ISCX VPN-nonVPN dataset 2016 for packet-level encrypted traffic classification can not incorporate packet header information, as it allows to directly map a packet to a specific application with high accuracy. Considering only non-VPN traffic, 95% of all packets in the dataset can be associated with an application. The remaining packets can still be classified with high probability by guessing based on the applications that use this flow. This result suggests that it is very likely that the Deep Packet approach proposed in [10] learns to memorize the mapping from flows to applications instead of finding application-specific encryption patterns.

While the ISCX VPN-nonVPN dataset might be suitable for statistical and timeseries approaches, future work should investigate real-world traffic in order to generate better datasets for packet-level classification.

Overall the situation in current research regarding encrypted traffic classification is methodically unsatisfying. Authors do not publish code, use non-public datasets, or do not explain data preprocessing steps in detail. This makes it hard to evaluate their results. Nonetheless, usage of statistical and timeseries data seems to be the state-of-the-art solution for encrypted traffic classification.

It is very unlikely that the Deep Packet approach actually learned how to find patterns in the encrypted data. It is more likey that the model learned to memorize the mapping from flows to applications.

Comparison

Technique	Advantage	Disadvantage
Port-based	Simple, fast, low resources	Detects mostly protocol level
DPI	Accurate, fine grain detection	Slow, high computational resources, unable to detect undefined signatures, privacy breaching
Supervised classification	Accurate, simple to implement, fine grain detection, fast	Performance depends on the trained dataset, feature engineering requirement
Unsupervised clustering	Do not require labelled dataset, fine grain detection	Lower performance than the supervised classification
Semi-supervised	Fine grain classification, higher accuracy than standalone supervised or clustering	Higher computational resources than supervised and supervised solutions
Deep learning	Feature engineering elimination, accurate, fine grain detection	The requirement of large amount of annotated data and computational resources

Dataset

ISCXVPN2016

Traffic: Content

Web Browsing: Firefox and Chrome

Email: SMPTS, POP3S and IMAPS

Chat: ICQ, AIM, Skype, Facebook and Hangouts

Streaming: Vimeo and Youtube

File Transfer: Skype, FTPS and SFTP using Filezilla and an external service

VoIP: Facebook, Skype and Hangouts voice calls (1h duration)

P2P: uTorrent and Transmission (Bittorrent)

ISCXTor2016

Traffic: Content

Web Browsing: Firefox and Chrome

Email: SMPTS, POP3S and IMAPS

Chat: ICQ, AIM, Skype, Facebook and Hangouts

Streaming: Vimeo and Youtube

File Transfer: Skype, FTP over SSH (SFTP) and FTP over SSL (FTPS) using Filezilla and an external service

VoIP: Facebook, Skype and Hangouts voice calls

P2P: uTorrent and Transmission (Bittorrent)

VPN/NONVPN NETWORK APPLICATION TRAFFIC DATASET (VNAT)

Traffic Category

Streaming: Vimeo, Netflix, Youtube

VoIP: Zoiper, Voip

Chat Skype

Command & Control (C2) SSH, RDP

File Transfer SFTP, RSYNC, SCP

Datasets References

- The ISCXVPN2016 dataset is publicly available for researchers. If you are using our dataset, you should cite our related research paper which outlining the details of the dataset and its underlying principles:

  Gerard Drapper Gil, Arash Habibi Lashkari, Mohammad Mamun, Ali A. Ghorbani, "Characterization of Encrypted and VPN Traffic Using Time-Related Features", In Proceedings of the 2nd International Conference on Information Systems Security and Privacy(ICISSP 2016) , pages 407-414, Rome, Italy.

- The ISCXTor2016 dataset is publicly available for researchers. If you are using our dataset, you should cite our related research paper which outlining the details of the dataset and its underlying principles:

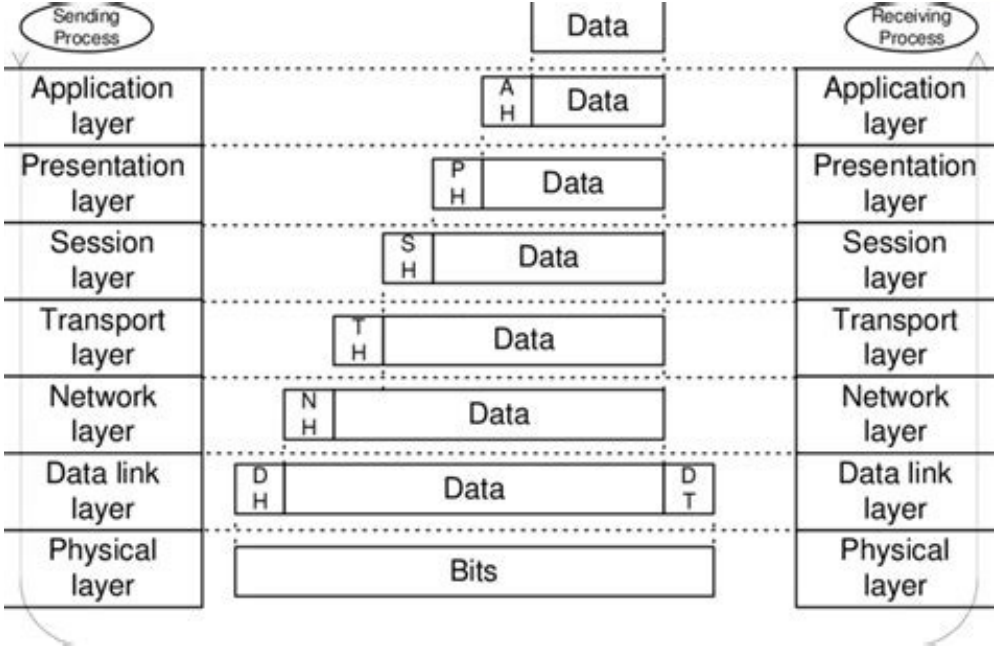
  Arash Habibi Lashkari, Gerard Draper-Gil, Mohammad Saiful Islam Mamun and Ali A. Ghorbani, "Characterization of Tor Traffic Using Time Based Features", In the proceeding of the 3rd International Conference on Information System Security and Privacy, SCITEPRESS, Porto, Portugal, 2017.

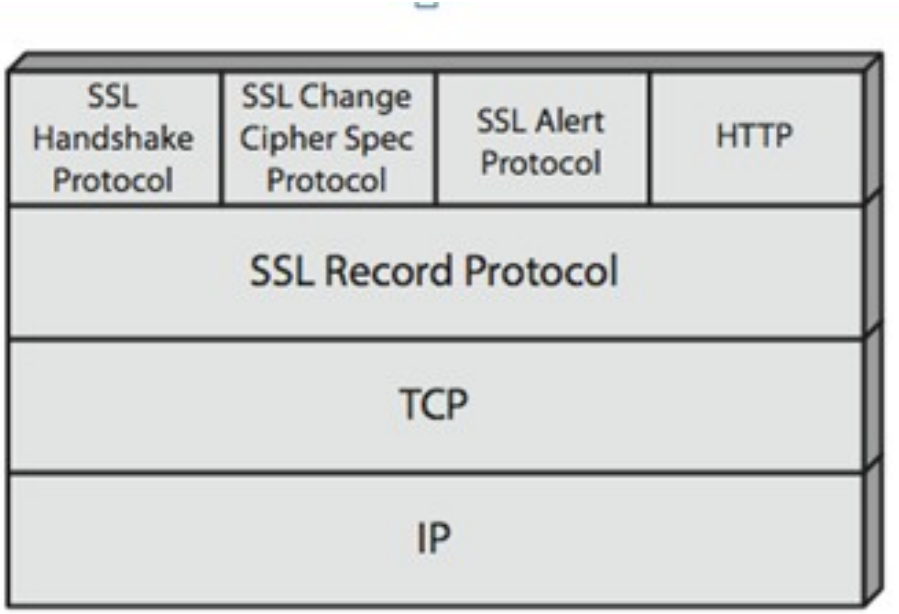
- S. Jorgensen et al., "Extensible Machine Learning for Encrypted Network Traffic Application Labeling via Uncertainty Quantification," in IEEE Transactions on Artificial Intelligence, doi: 10.1109/TAI.2023.3244168.

  Abstract: With the increasing prevalence of encrypted network traffic, cyber security analysts have been turning to machine learning (ML) techniques to elucidate the traffic on their networks. However, ML models can become stale as new traffic emerges that is outside of the distribution of the training set. In order to reliably adapt in this dynamic environment, ML models must additionally provide contextualized uncertainty quantification to their predictions, which has received little attention in the cyber security domain. Uncertainty quantification is necessary both to signal when the model is uncertain about which class to choose in its label assignment and when the traffic is not likely to belong to any pre-trained classes. We present a new, public dataset of network traffic that includes labeled, Virtual Private Network (VPN)-encrypted network traffic generated by 10 applications and corresponding to 5 application categories. We also present an ML framework that is designed to rapidly train with modest data requirements and provide both calibrated, predictive probabilities as well as an interpretable “out-of-distribution” (OOD) score to flag novel traffic samples. We describe calibrating OOD scores using p-values of the relative Mahalanobis distance. We demonstrate that our framework achieves an F1 score of 0.98 on our dataset and that it can extend to an enterprise network by testing the model: (1) on data from similar applications, (2) on dissimilar application traffic from an existing category, and (3) on application traffic from a new category. The model correctly flags uncertain traffic and, upon retraining, accurately incorporates the new data.

  URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10044382&isnumber=9184921

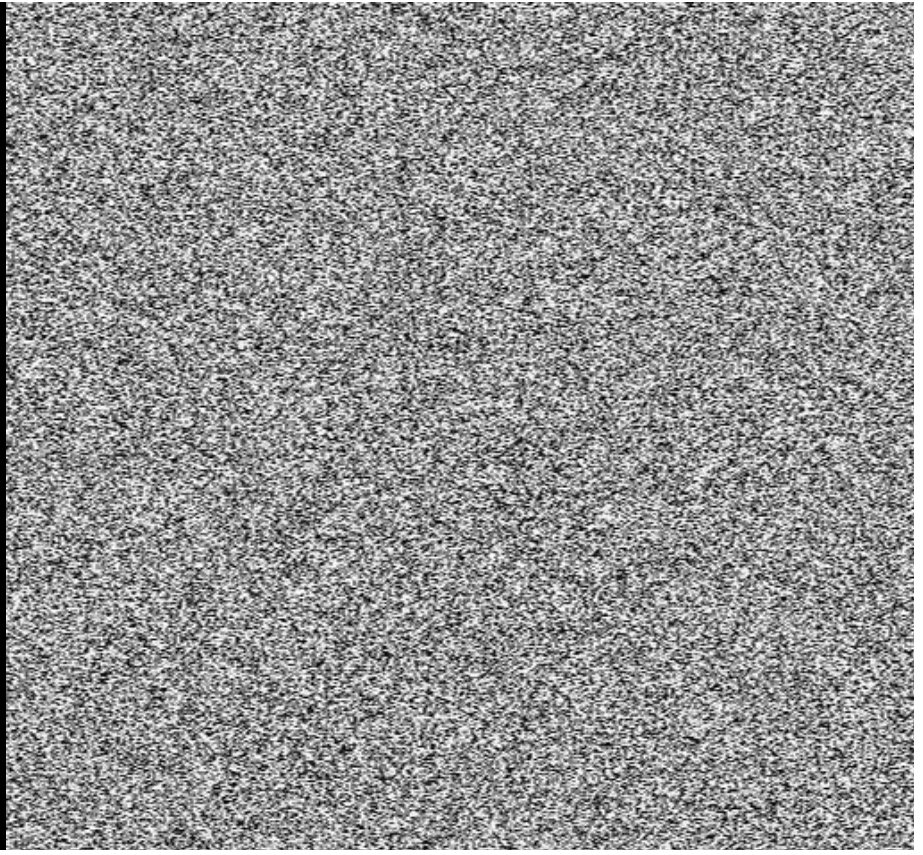
Ncap data


Sending Process
Receiving Process
Data
Application layer
A H
Data
Presentation layer
P H
Data
Session layer
S H
Data
Transport layer
T H
Data
Network layer
N H
Data
Data link layer
D H
Data
D T
Physical layer
Bits


SSL Handshake Protocol
SSL Change Cipher Spec Protocol
SSL Alert Protocol
HTTP
SSL Record Protocol
TCP
IP

Ncap data

```
0000000 0000 0001 0001 1010 0010 0001 0004 0128
0000010 0000 0016 0000 0028 0000 0010 0000 0020
0000020 0000 0001 0004 0000 0000 0000 0000 0000
0000030 0000 0000 0000 0010 0000 0000 0000 0204
0000040 0004 8384 0084 c7c8 00c8 4748 0048 e8e9
0000050 00e9 6a69 0069 a8a9 00a9 2828 0028 fdfc
0000060 00fc 1819 0019 9898 0098 d9d8 00d8 5857
0000070 0057 7b7a 007a bab9 00b9 3a3c 003c 8888
0000080 8888 8888 8888 8888 288e be88 8888 8888
0000090 3b83 5788 8888 8888 7667 778e 8828 8888
00000a0 d61f 7abd 8818 8888 467c 585f 8814 8188
00000b0 8b06 e8f7 88aa 8388 8b3b 88f3 88bd e988
00000c0 8a18 880c e841 c988 b328 6871 688e 958b
00000d0 a948 5862 5884 7e81 3788 1ab4 5a84 3eec
00000e0 3d86 dcb8 5cbb 8888 8888 8888 8888 8888
00000f0 8888 8888 8888 8888 8888 8888 8888 0000
0000100 0000 0000 0000 0000 0000 0000 0000 0000
*
0000130 0000 0000 0000 0000 0000 0000 0000
000013e
```



Why not to use known headers?

Combining datasets

- Used library: scapy
- Used method: merge and adapt timestamps
- Also merged CIC-* files with pandas.
- Datasets are bigger then 100G.

  So this would be a part 2:)

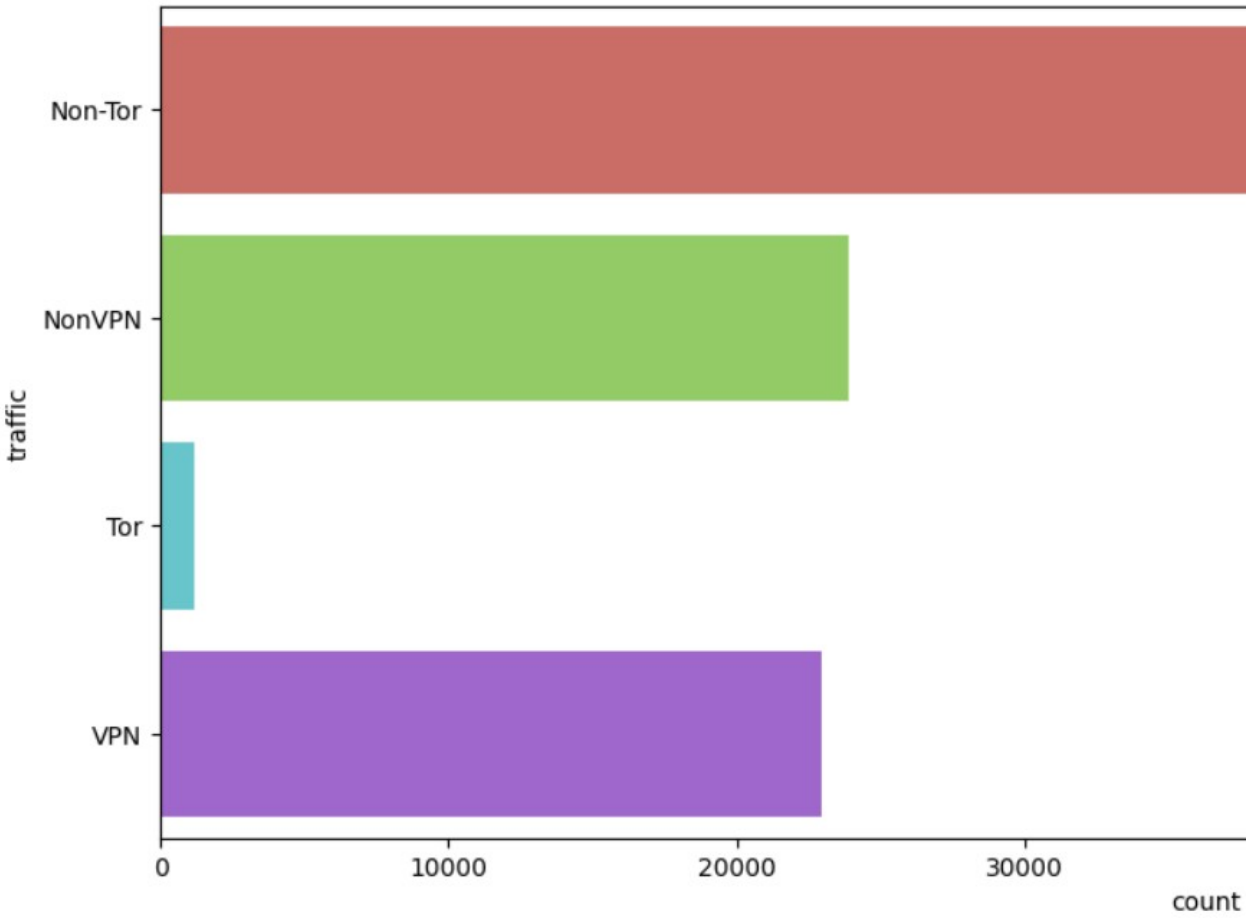
```python
import os
import argparse
from scapy.all import rdpcap, wrpcap

def merge_pcap_files(input_directory, output_filename):
    pcap_files = [f for f in os.listdir(input_directory) if f.endswith('.pcap')]

    packets = []

    for pcap_file in pcap_files:
        file_path = os.path.join(input_directory, pcap_file)
        print(f"Reading packets from {file_path}")
        pcap = rdpcap(file_path)

        if packets:
            time_shift = packets[-1].time - pcap[0].time
            for pkt in pcap:
                pkt.time += time_shift

        packets.extend(pcap)

    output_path = os.path.join(input_directory, output_filename)
    print(f"Writing merged packets to {output_path}")
    wrpcap(output_path, packets)
    print("Merge complete.")

if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='Merge pcap files in a directory.')
    parser.add_argument('input_directory', help='Path to the directory containing pcap files')
    parser.add_argument('output_filename', help='Name of the output merged pcap file')
    args = parser.parse_args()
    merge_pcap_files(args.input_directory, args.output_filename)
~
```

Predicting classes without payload (wrong)

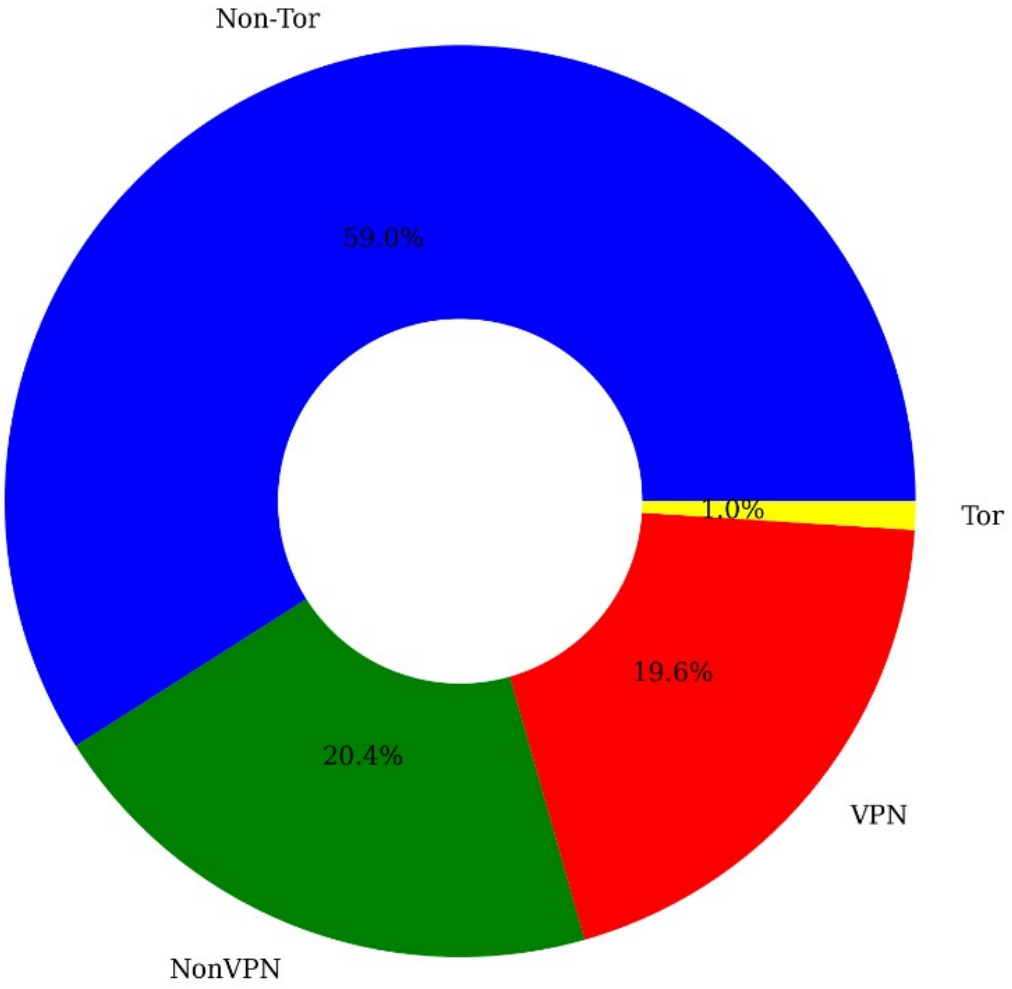
```
Resulting dataset shape
(141530, 85)
```

	Flow ID	Src IP	Src Port	Dst IP	Dst Port	Protocol	Timestamp	Flow Duration	Total Fwd Packet	Total Bwd packets	...
0	10.152.152.11-216.58.220.99-57158-443-6	10.152.152.11	57158	216.58.220.99	443	6	24/07/2015 04:09:48 PM	229	1	1	...
1	10.152.152.11-216.58.220.99-57159-443-6	10.152.152.11	57159	216.58.220.99	443	6	24/07/2015 04:09:48 PM	407	1	1	...
2	10.152.152.11-216.58.220.99-57160-443-6	10.152.152.11	57160	216.58.220.99	443	6	24/07/2015 04:09:48 PM	431	1	1	...
3	10.152.152.11-74.125.136.120-49134-443-6	10.152.152.11	49134	74.125.136.120	443	6	24/07/2015 04:09:48 PM	359	1	1	...
4	10.152.152.11-173.194.65.127-34697-19305-6	10.152.152.11	34697	173.194.65.127	19305	6	24/07/2015 04:09:45 PM	10778451	591	400	...

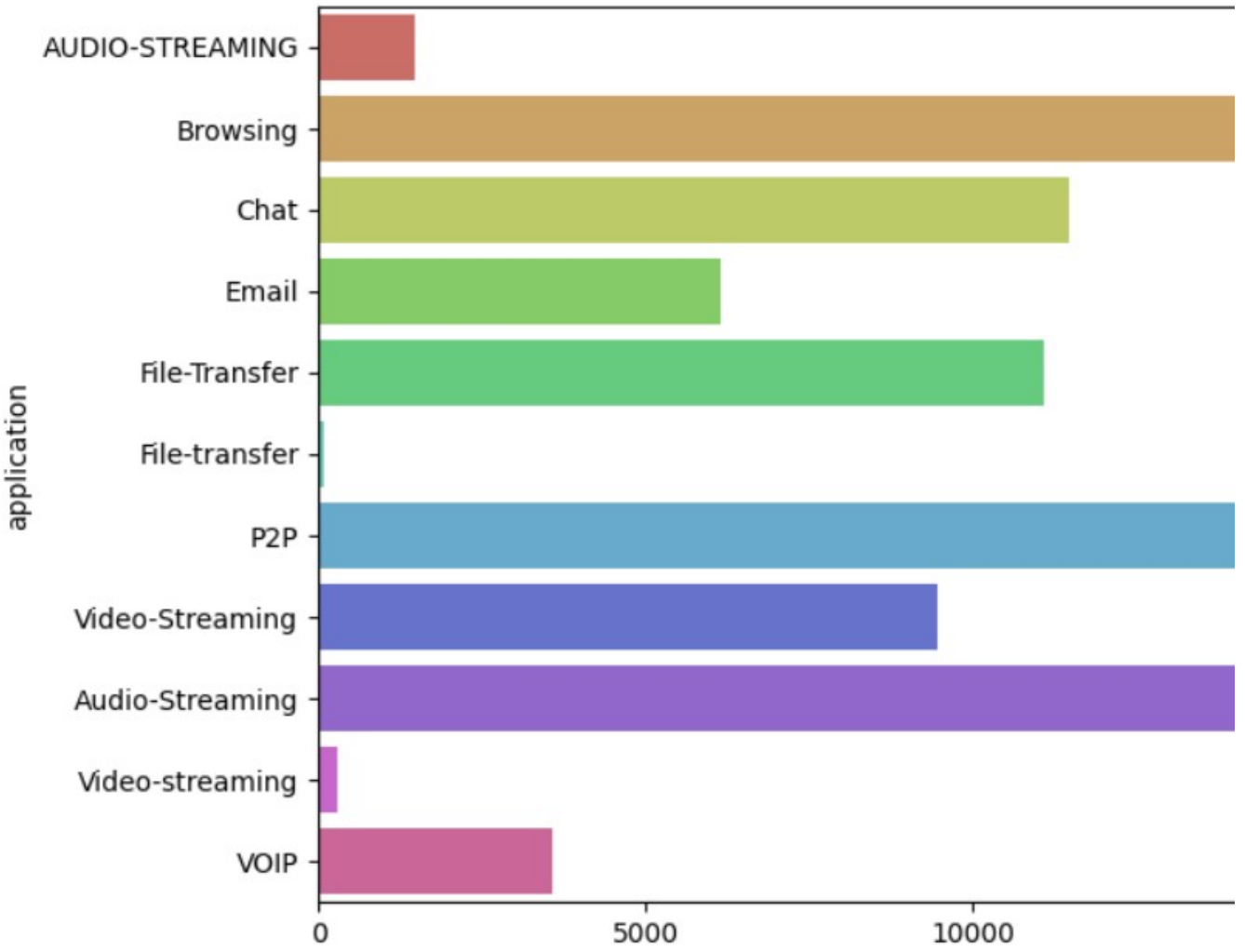
Predicting classes without payload (wrong)



Predicting classes without payload (wrong)



Predicting classes without payload (wrong)



Predicting classes without payload (wrong)

Splitting the data

```
NOT_VALUABLE_COLS = ['Src IP', 'Flow ID', 'Dst IP', 'Timestamp']

input_data = combined_df.drop(['traffic','application'] + NOT_VALUABLE_COLS, axis=1)
traffic_label = combined_df['traffic']
application_label = combined_df['application']
# Split data for the traffic label
Xtrain_traffic, Xtest_traffic, ytrain_traffic, ytest_traffic = train_test_split(
    input_data,traffic_label , random_state=104,test_size=0.2, shuffle=True)

Xtrain_traffic, Xval_traffic, ytrain_traffic, yval_traffic = train_test_split(
  Xtrain_traffic,ytrain_traffic , random_state=104,test_size=0.25, shuffle=True)

# Split data for the application label
Xtrain_application, Xtest_application, ytrain_application, ytest_application = train
  input_data,application_label , random_state=104,test_size=0.2, shuffle=True)

Xtrain_application, Xval_application, ytrain_application, yval_application = train_t
  Xtrain_application,ytrain_application , random_state=104, test_size=0.25, shuffle=
```

Predicting classes without payload (wrong)

Performed data preprocessing:

- Dropped N/A and +-inf columns
- Reduced dims with VarianceThreshold
- Reduced dims with RandomForest as Feature selector
- Dropped some columns manually

Predicting classes without payload (wrong)

Performed Model training:

- Trained RF
- **Separate** model for **traffic**, and **separate** for **application**
- GridSearchCV with StratifiedKFold (4 folds)
- balanced accuracy as ef
- param_grid = *{*

'n_estimators': [50, 150, 300],

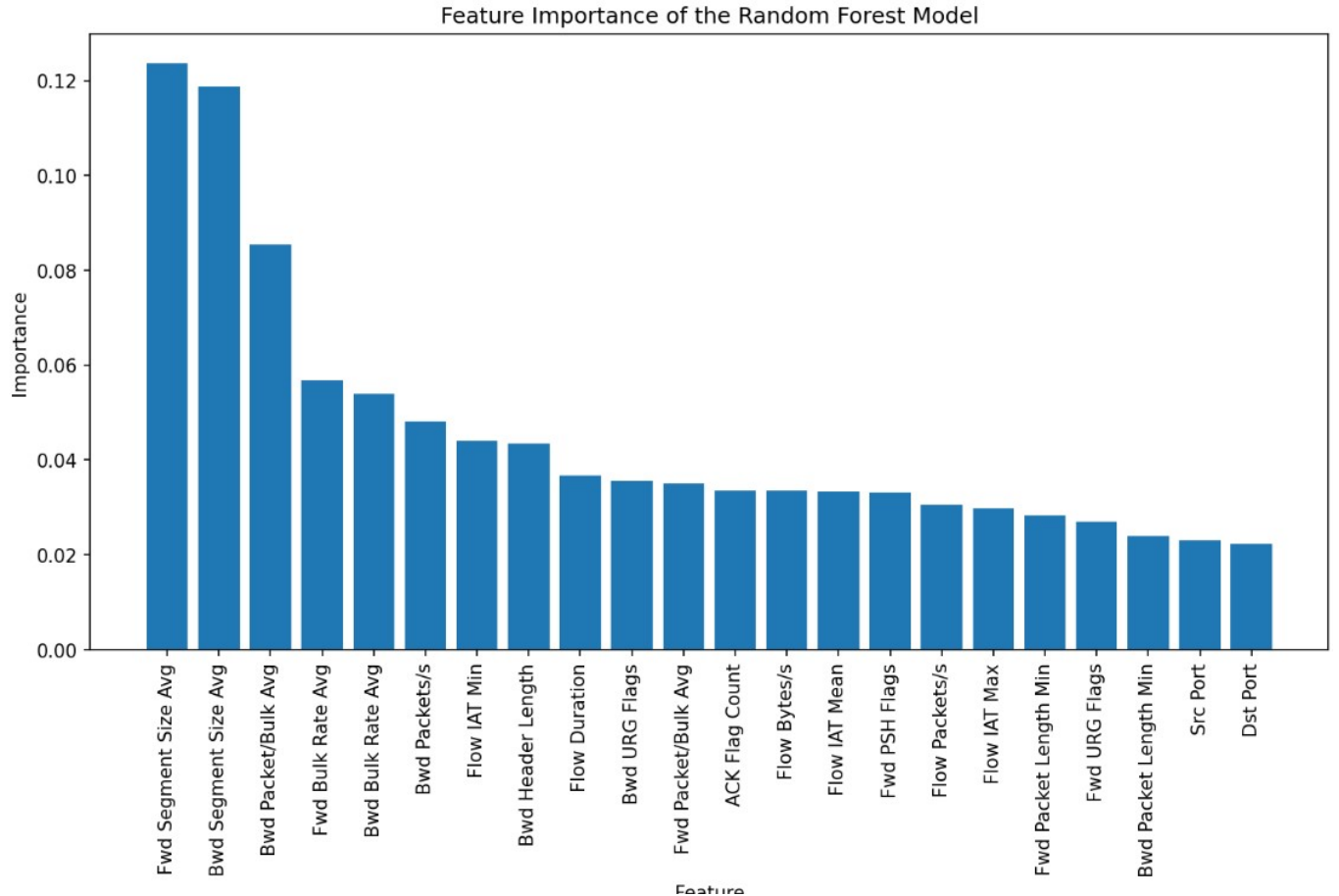
'max_depth': [None, 10, 30],

}

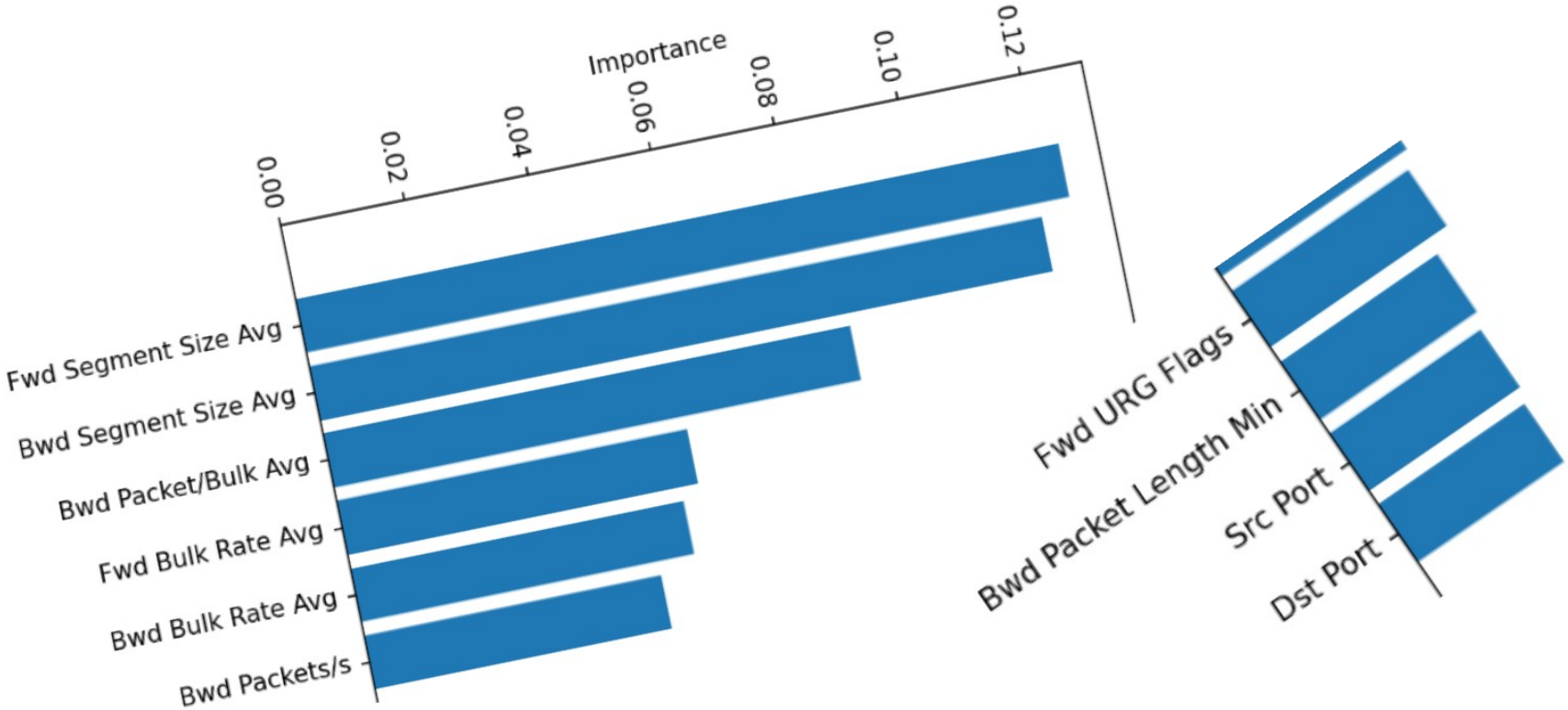
‘Traffic’ prediction results: classification report

	precision	recall	f1-score	support
Non-Tor	1.00	1.00	1.00	13870
NonVPN	0.95	0.96	0.96	4729
Tor	0.95	0.89	0.92	258
VPN	0.96	0.95	0.96	4548
accuracy			0.98	23405
macro avg	0.97	0.95	0.96	23405
weighted avg	0.98	0.98	0.98	23405

‘Traffic’ prediction results: feature importance



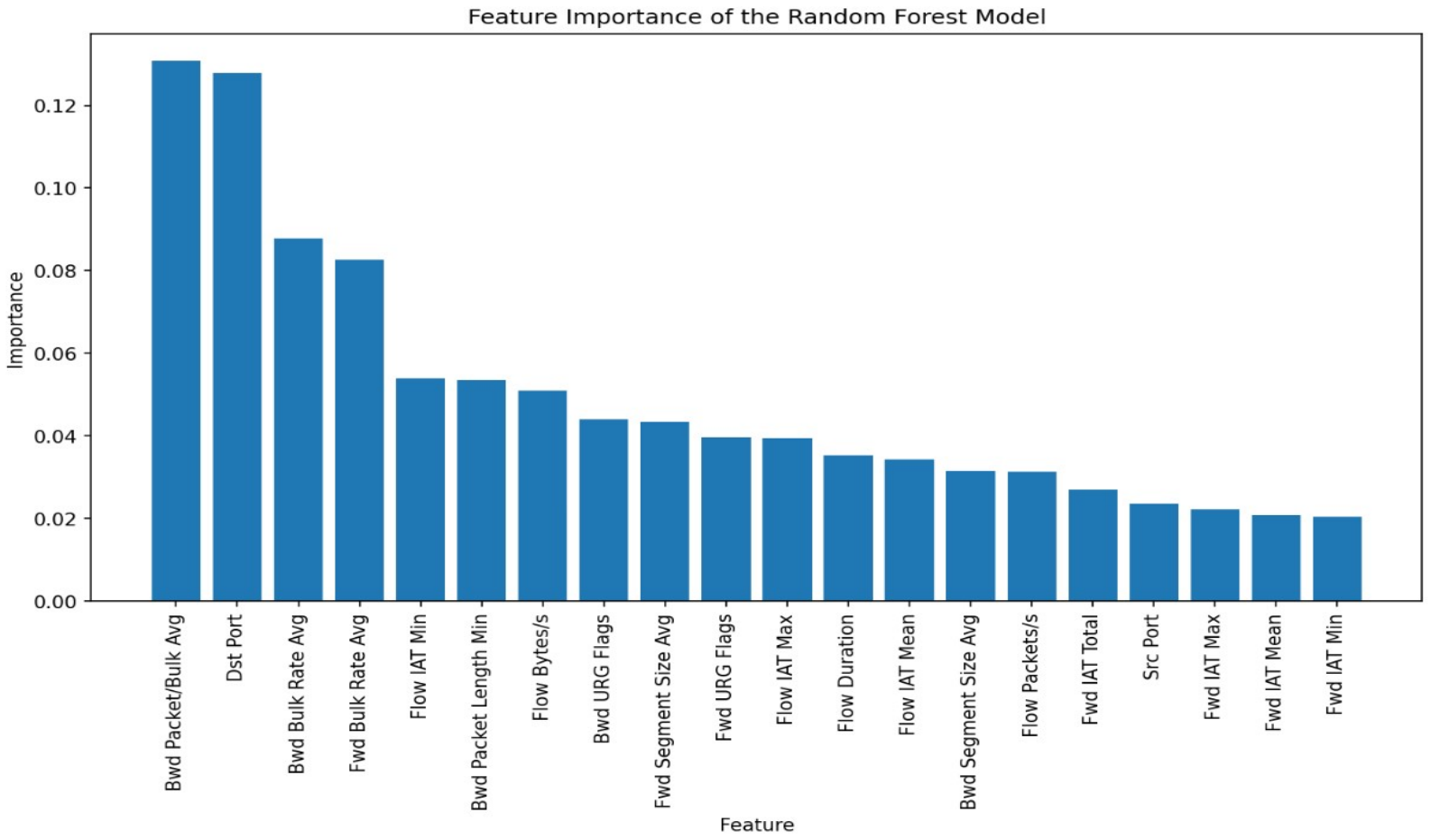
‘Traffic’ prediction results: feature importance



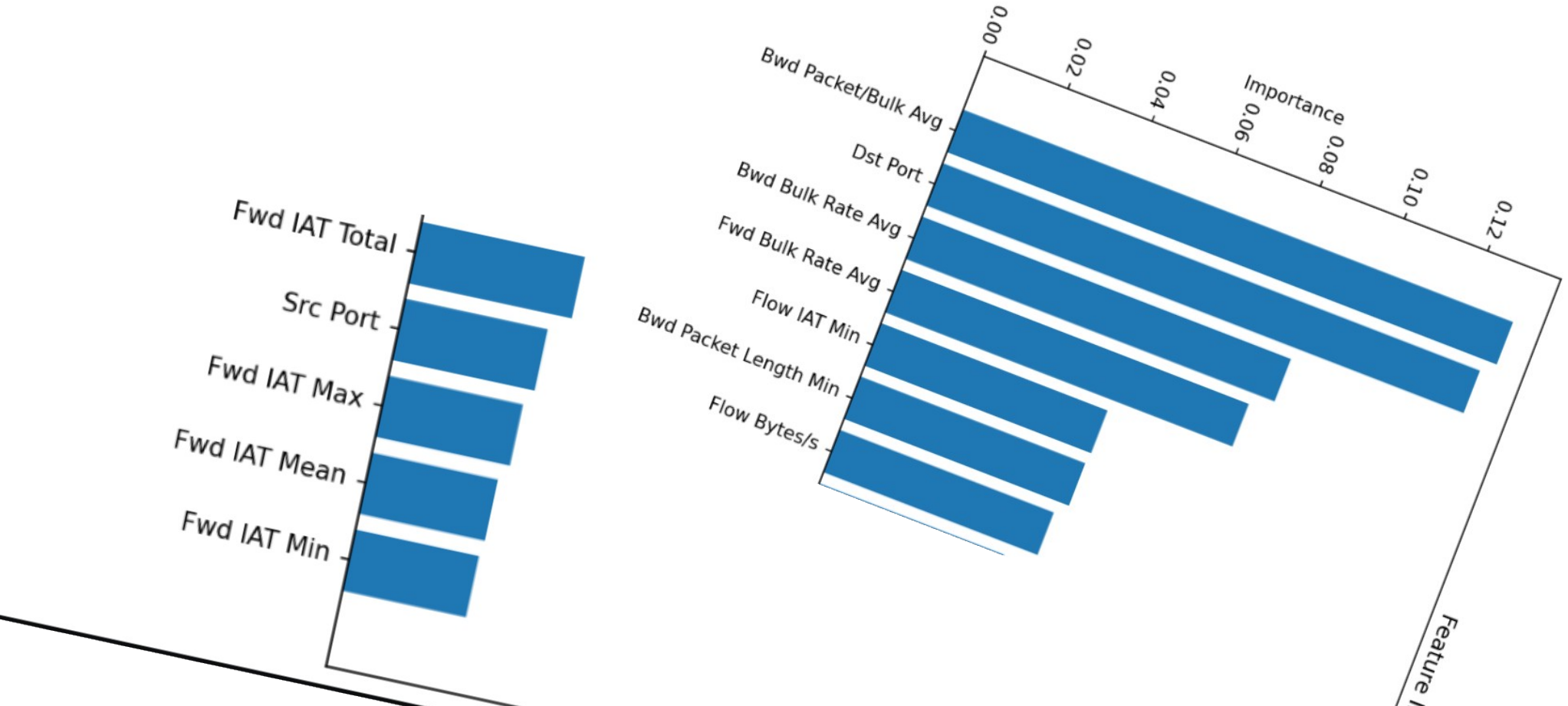
‘Application’ prediction results: classification report

	precision	recall	f1-score	support
Non-Tor	1.00	1.00	1.00	13870
NonVPN	0.95	0.96	0.96	4729
Tor	0.95	0.89	0.92	258
VPN	0.96	0.95	0.96	4548
accuracy			0.98	23405
macro avg	0.97	0.95	0.96	23405
weighted avg	0.98	0.98	0.98	23405

‘Application’ prediction results: feature importance



‘Application’ prediction results: feature importance



Conclusion

- Feature selection and understanding is very important.
- Results are positive but useless. But could get better.
- There are **complications:**
  - Randomization.
  - Masquerading.
  - Encryption.
  - Dataset collection and labeling..
  - Imbalance dataset.
  - Adversarial example.

Future work

1) Get more data (quantity and diversity)
2) Train CNN on the binary payloads
3) Try to generate own dataset