Assignment 2

For due date see learn.bcit.ca

All work should be done individually.

Deliverables

- The final `NIDS.py` script along with any required files necessary for it to load up the trained classifiers.
- A report (in `.pdf` format) that describes how you did the assignment; in particular, explain your feature selection technique, the selected classification frameworks and their performance comparison. Put the classification scores in the report as well. Don't forget to mention the problems you had in implementation of your code (if any). Also explain any missing components of your implementation as well as your thoughts on the problem(s).

Intelligent Network Intrusion Detection Agent

This assignment describes the data and requirement for you to create an intelligent network intrusion detection agent. To get familiar with the topic, first, lets read the following text from [1]:

> Currently, due to the massive growth in computer networks and applications, many challenges arise for cyber security systems. Intrusions/attacks can be defined as a set of events which are able to compromise the principles of computer systems, e.g. availability, authority, confidentiality and integrity. Firewall systems cannot detect modern attack environments and are not able to analyse network packets in depth. Because of these reasons, Intrusions Detection Systems are designed to achieve high protection for the cyber security infrastructure.
>
> A Network Intrusion Detection System (NIDS) monitors network traffic flow to identify attacks. NIDSs are classified into misuse/signature and anomaly based. The signature based matches the existing of known attacks to detect intrusions. However, in the anomaly based, a normal profile is created from the normal behavior of the network, and any deviation from this is considered as attack. Further, the signature based NIDSs cannot detect unknown attacks, and for these anomaly NIDS are recommended in many studies.
>
> The effectiveness of NIDS is evaluated based on their performance to identify attacks which requires a comprehensive data set that contains normal and abnormal behaviors.

Researchers at the University of New South Wales (UNSW; Australia) have collected 100 GB of the raw network traffic using the IXIA PerfectStorm tool and created *UNSW-NB15*

dataset containing 49 features with the class label indicating whether the record is normal or not (Table 1 provides the full list of the extracted features). The dataset is *a hybrid of real modern normal activities and synthetic contemporary attack behaviours*, and has nine types of attacks, namely, **Fuzzers**, **Analysis**, **Backdoors**, **DoS**, **Exploits**, **Generic**, **Reconnaissance**, **Shellcode** and **Worms**. You are highly recommended to read the full paper explaining the data and how it was generated to get more familiar with it (a copy of the paper is provided to you with this file).

Since the full dataset is quite big (and unbalanced), you work on a subset of the dataset which is balanced and smaller. The `UNSW-NB15-BALANCED-TRAIN.csv` data file that you receive follows the same composition rules as the full data except that it has 449,797 training records.

Assignment Description

Your task is to read about the data and create a `NIDS.py` script which receives the name of a test file (with the exact same format as `UNSW-NB15-BALANCED-TRAIN.csv`) as well as a classifier name (e.g. it should be callable with the command '`python NIDS.py heldout_testset.csv your_selected_classification_method_name`'), classifies the network traffic records in it and produces the proper classification reports (**If your script does not run this way your submission will not be graded and you will get a 0!**). In each task you will consider two separate class labels `Label` (predicting whether the record is normal or not) and `attack_cat` (predicting the attack category). **Clearly, you are not allowed to use `attack_cat` and `Label` attributes as input features to any of your models**. However, you will use those categories to select proper features in the first step.

Feature Analysis/Selection

1. Choose 3 different analysis techniques[1].
2. Explain how each technique can help in feature selection.
3. Provide charts/graphs as well as possible extracted scores/stats using the technique and finalize your selected feature set for performing the classification task.
4. Take one of the classifiers from the next section and compare its classification scores with and without feature selection applied to the data.
5. Spend at most two pages per technique in your report.

[1] You can choose from different techniques such as feature co-variance/correlation analysis, recursive feature elimination, principal component analysis, entropy based feature importance analysis, ... Just make sure the analysis types are different.

- **Do not** throw in tens of graphs without any explanation, your explanation matters more to me than the graphs!
- **Do not** put the analysis code in the report, just explain your findings and point me to the proper source code file/line in your submission.

`Label` Classification

As you notice, you are not provided with a validation set. You may want to reserve a chunk of the provided training data to perform the internal testing and the model selection. Make sure your model is not using your validation set as training data.

Choose **3 different classification techniques** (you can simply choose the pre-implemented sklearn classifiers) and report label classification scores for each. Your submitted report should contain three formatted tables looking like the following:

Classifier: <classifier–name>

	precision	recall	f1–score	support
0	0.00	0.00	0.00	00000
1	0.00	0.00	0.00	00000
accuracy			0.00	00000
macro avg	0.00	0.00	0.00	00000
weighted avg	0.00	0.00	0.00	00000

Report which one worked the best and explain why it worked better than the others.

All three classifiers you submit must report F1 scores above 0.9. I will also test your submission with my held-out test set and expect your submission to get F1 scores above 0.85. Any of the models not passing these criteria **will not receive a grade**.

`attack_cat` Classification

The main classification task in this report is concerned with classification of attack categories. As you might have noticed, the binary classification of `Label` attribute is quite an easy task and you can get high accuracy scores (why should this happen?). In this section, you will try classifying the records which are labelled as an attack. Our objective is to get the **highest Macro-F1** classification score. Like previous part choose **3 different classification techniques** and report label classification scores for each. Make sure you report both Micro-F1 and Macro-F1 scores.
Your submitted report should contain three formatted tables looking like the following:

Classifier: <classifier–name>

	precision	recall	f1–score	support
None	0.00	0.00	0.00	00000
Generic	0.00	0.00	0.00	00000
Fuzzers	0.00	0.00	0.00	00000
Exploits	0.00	0.00	0.00	00000
DoS	0.00	0.00	0.00	00000
Fuzzers	0.00	0.00	0.00	00000
Reconnaissance	0.00	0.00	0.00	00000
Backdoor	0.00	0.00	0.00	00000
Reconnaissance	0.00	0.00	0.00	00000
Analysis	0.00	0.00	0.00	00000
Shellcode	0.00	0.00	0.00	00000
Shellcode	0.00	0.00	0.00	00000
Backdoors	0.00	0.00	0.00	00000
Worms	0.00	0.00	0.00	00000
micro avg	0.00	0.00	0.00	00000
macro avg	0.00	0.00	0.00	00000
weighted avg	0.00	0.00	0.00	00000

In the report explain what is the main issue in the dataset that the classifiers would suffer from and what was your approach to deal with it.

All three classifiers you submit must report Macro-F1 scores above 0.45. I will also test your submission with my held-out test set and expect your submission to get Macro-F1 scores above 0.40. Micro-F1 should also be above 0.9 on the validation set and above 0.85 on my held-out test set. Any of the models not passing these criteria **will not receive a grade**.

A few tips to do this part:

- Don't forget that in the provided train data file, `attack_cat` is empty for non-attack records!
- You may use the prediction result of your `Label` classifier to reduce the complexity of multi-class classification for `attack_cat` classifier.

References

[1] Moustafa, N., & Slay, J. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In 2015 military communications and information systems conference (MilCIS) (pp. 1–6). Can be found under https://www.researchgate.net/profile/Nour-Moustafa/publication/

Appendix

Table 1: The description of the extracted features as explained in UNSW-NB15_features.csv file.

Name	Type	Description
srcip	nominal	Source IP address
sport	integer	Source port number
dstip	nominal	Destination IP address
dsport	integer	Destination port number
proto	nominal	Transaction protocol
state	nominal	Indicates to the state and its dependent protocol, e.g. ACC, CLO, CON, ECO, ECR, FIN, INT, MAS, PAR, REQ, RST, TST, TXD, URH, URN, and (-) (if not used state)
dur	Float	Record total duration
sbytes	Integer	Source to destination transaction bytes
dbytes	Integer	Destination to source transaction bytes
sttl	Integer	Source to destination time to live value
dttl	Integer	Destination to source time to live value
sloss	Integer	Source packets retransmitted or dropped
dloss	Integer	Destination packets retransmitted or dropped
service	nominal	http, ftp, smtp, ssh, dns, ftp-data ,irc and (-) if not much used service
Sload	Float	Source bits per second
Dload	Float	Destination bits per second
Spkts	integer	Source to destination packet count
Dpkts	integer	Destination to source packet count
swin	integer	Source TCP window advertisement value
dwin	integer	Destination TCP window advertisement value
stcpb	integer	Source TCP base sequence number
dtcpb	integer	Destination TCP base sequence number
smeansz	integer	Mean of the flow packet size transmitted by the src
dmeansz	integer	Mean of the flow packet size transmitted by the dst
trans_depth	integer	Represents the pipelined depth into the connection of http request/response transaction
res_bdy_len	integer	Actual uncompressed content size of the data transferred from the server's http service.
Sjit	Float	Source jitter (mSec)
Djit	Float	Destination jitter (mSec)

Stime	t_stamp	record start time
Ltime	t_stamp	record last time
Sintpkt	Float	Source interpacket arrival time (mSec)
Dintpkt	Float	Destination interpacket arrival time (mSec)
tcprtt	Float	TCP connection setup round-trip time, the sum of 'synack' and 'ackdat'.
synack	Float	TCP connection setup time, the time between the SYN and the SYN_ACK packets.
ackdat	Float	TCP connection setup time, the time between the SYN_ACK and the ACK packets.
is_sm_ips_ports	Binary	If source (1) and destination (3)IP addresses equal and port numbers(2)(4) equal then, this variable takes value 1 else 0
ct_state_ttl	Integer	No. for each state (6) according to specific range of values for source/destination time to live (10) (11).
ct_flw_http_mthd	Integer	No. of flows that has methods such as Get and Post in http service.
is_ftp_login	Binary	If the ftp session is accessed by user and password then 1 else 0.
ct_ftp_cmd	integer	No of flows that has a command in ftp session.
ct_srv_src	integer	No. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26).
ct_srv_dst	integer	No. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26).
ct_dst_ltm	integer	No. of connections of the same destination address (3) in 100 connections according to the last time (26).
ct_src_ ltm	integer	No. of connections of the same source address (1) in 100 connections according to the last time (26).
ct_src_dport_ltm	integer	No of connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26).
ct_dst_sport_ltm	integer	No of connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26).
ct_dst_src_ltm	integer	No of connections of the same source (1) and the destination (3) address in in 100 connections according to the last time (26).

attack_cat	nominal	The name of each attack category. In this data set , nine categories e.g. Fuzzers, Analysis, Backdoors, DoS Exploits, Generic, Reconnaissance, Shellcode and Worms
Label	binary	0 for normal and 1 for attack records