Student Name : Cholakov Kristiyan Kamenov

Group : FDDP1

Date : 18.10.2022

LAB 4: ANALZING NETWORK DATA LOG

You will be provided with the data file, in .csv format, in the working directory. Write the program to extract the following informations.

EXERCISE 4A: TOP TALKERS AND LISTENERS

One of the most commonly used function in analyzing data log is finding out the IP address of the hosts that send out large amount of packet and hosts that receive large number of packets, usually know as TOP TALKERS and LISTENERS. Based on the IP address we can obtained the organization who owns the IP address.

TOP 5 TALKERS

Rank	IP address	# of packets	Organisation
1	193.62.192.8	3041	JISC Services Limited
2	155.69.160.32	2975	Nanyang Technological University
3	130.14.250.11	2604	National Library of Medicine
4	14.139.196.58	2452	NKN EDGE Network
5	140.112.8.139	2056	Taiwan Academic Network

TOP 5 LISTENERS

Rank	IP address	# of packets	Organisation
1	103.37.198.100	3841	A*STAR
2	137.132.228.15	3715	NUS
3	202.21.159.244	2446	Republic Polytechnic
4	192.101.107.153	2368	Battelle Memorial Institute
5	103.21.126.2	2056	Powai

EXERCISE 4B: TRANSPORT PROTOCOL

Using the IP protocol type attribute, determine the percentage of TCP and UDP protocol

	Header value	Transport layer protocol	# of packets	%
1	6	TCP	56064	80.818798
2	17	UDP	9462	13.639902
3	50	ESP	1698	2.447744
4	0	HOPOPT	1261	1.817789
5	47	GRE	657	0.947095
6	41	IPv6	104	0.149921
7	1	ICMP	74	0.106674
8	381	Header value should be in 0-255 range	45	0.064870
9	58	IPv6-ICMP	4	0.005766
10	103	PIM	1	0.001442

EXERCISE 4C: APPLICATIONS PROTOCOL

Using the Destination IP port number determine the TOP 5 most frequently used application protocol.

Rank	Destination IP port number	# of packets	Service
1	443	13423	HTTPS
2	80	2647	HTTP
3	52866	2068	Dynamic
4	45512	1356	Unassigned
5	56152	1341	Dynamic

EXERCISE 4D: TRAFFIC INTENSITY

The traffic intensity is an important parameter that a network engineer needs to monitor closely to determine if there is congestion. You would use the IP packet size to calculate the estimated total traffic over the monitored period of 15 seconds. (The sampling rate is as givenin the data provided. Possibly 1 in 2048)

Total calculated sampled traffic (MB): 61.77694511413574 MB (calculated with MB = 2^{20} bytes)

Estimated Total Traffic taking into account the sampling rate (MB)	126519.18359375 MB (calculated with MB = 2^{20} bytes)

EXERCISE 4E: ADDITIONAL ANALYSIS

Top 5 Connection Pairs

Rank	Address 1	Organization 1	Address 2	Organization 2	# of Packets
1	137.132.228.15	NUS	193.62.192.8	European Bioinformatics	4951
2	103.37.198.100	A*STAR	130.14.250.11	National Library of Medicine	2842
3	14.139.196.58	Indian Institute of Technology	192.101.107.153	Battelle Memorial Institute	2368
4	140.112.8.139	Taiwan Academic Network	103.21.126.2	Powai	2056
5	140.90.101.61	National Oceanic and Atmospheric Administration	167.205.52.8	Institut Teknologi Bandung	1752

Based on the results from the top 5 connection pars, we can consider that the sampled data may be from a router from Singapore because the NUS and A*STAR figure in the top 2 connection. Also, most of the organizations from the top 5 connection pairs are educational and research institutions. Thus, we may assume that the sampled network is used for educational/research purposes.

IP Geolocation

Talkers

This map represents the number of talkers from the sampled data plotted on the world map. The bubble shapes are used to give an idea of the volume of talkers in each country. We can see that the countries with most talkers are Singapore, USA, China, India, etc. Following this and the data from the top connection pairs, we can say that the singaporean educational and research institutions communicate with many foreighn ones .

Listeners

This map represents the same data but for the number of listeners in the countries. The difference is maybe that the number of listeners outside of Singapore is closer to the number of listeners in Singapore. Meaning the difference between Singaporean and foreign listeners is not so significant. Overall, the conclusion is again the same.

The above bar charts show that in fact the difference between the number of listeners in Singapore and the foreign ones is less than the one for the talkers

Talkers

Listeners

These maps show the actual locations of the talkers and listeners from the sampled data across the globe.

<u>Network Geo Map</u>

The world map shows the interactions between the nodes in the top 10 connection pairs (bidirectional) with most packets sent between them. The red paths represent the path where each end is one of the two nodes from the connection pair. The graph further proves that the main traffic in the sampled network is between Singapore and some other foreign countries like UK, USA and China.

Network Graph

Now lets explore the relations between the single-direction pairs with most packets sent between them.

First, we will do the plotting based on the top 5% of the pairs.

This part of the graph represents the connections with most packets sent between NTU and other nodes. We can say that because the IP address (155.69.160.32) of the node that combines most of the other nodes in one graph is an NTU's one.

As for this one, we can say this segment shows the interaction between UK's Science Technology Facilities and Taiwan Academic Network as most of the IP addresses in the graph (130.246.176.41 and 140.115.32.82) are registered to these organizations.

We can also, show the top 1% of the pairs with most packets sent between them. However, the nodes here are significantly less than in the graph about the top 5%. Thus, not many interactions between the different nodes can be monitored.

Port Scanning

src_IP	dst_IP	
193.62.192.8	137.132.228.15	1447
137.132.228.15	193.62.192.8	1009
0	0	90
59.191.192.73	130.194.11.52	90
203.117.152.208	103.37.196.23	81
74.125.10.59	123.136.64.29	68
198.71.44.98	152.3.219.19	67
207.241.228.123	103.18.0.34	66
140.90.101.61	167.205.52.8	61
152.3.219.19	198.71.44.98	60
167.205.52.8	140.90.101.61	59
202.21.158.254	175.156.15.109	53
4	0	45
20456354	1	45
202.21.158.254	124.197.82.28	40
198.71.44.169	216.165.102.1	39
13.107.4.50	202.21.158.37	38
31.13.78.52	202.21.159.241	36
140.117.69.1	103.18.0.34	36
194.254.137.126	103.237.142.250	31

If we group all entries by their source and destination IP addresses, we will have all the single-directional pairs in the sampled network. Then by counting the unique destination ports we can check which listener's ports a talker tries to access.

From the table on the left, we can see that talker node with IP address: 193.62.192.8 sends packets ot 1447 different ports of listener node with IP address: 137.132.228.15. Also, vice-versa packets are transmitted to 1009 unique ports.

If we check the organization to which the IP addresses belong, we will see that 193.62.192.8 is registered under "European Bioinformatics", while 137.132.228.15 belongs to "NUS".

Further, if we plot the different listener's ports that are accessed on a scatter plot, we can see that NUS's node sends packets to EB's ports in the range from 0 to ~ 65 000. With 65,535 being the maximum port number, it seems like NUS's node is doing some kind of port scanned to EB. We can see the same thing vice-versa but EB sends to ports in the interval 35 000 – 60 000. Both connections access ports in the range 49 151 – 65 536 which are dynamic or private and can be accessed by basically everyone.

These two connections are the only conncections where these phenomenon can be spotted. Also, the scanning is between only two nodes, none of the nodes does the same thing with other nodes from the sampled data.

EXERCISE 4F: SOFTWARE CODE

The next pages represent the Jupyter Notebook file and some of the graphs in form of images with higher resolution.

Importing the required libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
from pandas import *
from numpy import *
from matplotlib.pyplot import *
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.metrics import *
from sklearn.tree import *
from sklearn.ensemble import *
from sklearn.naive_bayes import *
from sklearn.neighbors import KNeighborsClassifier, RadiusNeighborsClassifier
from sklearn.svm import *
from sklearn.linear_model import *
from sklearn.preprocessing import *
from sklearn.cluster import *
from sklearn.manifold import *
from sklearn.metrics import pairwise_distances
from sklearn.datasets import *
from sklearn.decomposition import *
from sklearn.interactive import *
from sklearn.decomposition import PCA, KernelPCA
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.model_selection import cross_val_score
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression
from mlxtend.preprocessing import DenseTransformer
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
from sklearn import tree
from sklearn import ensemble
from sklearn import neighbors
from sklearn import svm
from sklearn import linear_model
from sklearn import naive_bayes
from sklearn import metrics
from sklearn import model_selection
from sklearn import preprocessing
from sklearn import decomposition
from sklearn import manifold
from sklearn import cluster
from sklearn import datasets
from sklearn import interactive
from sklearn import pipeline
from sklearn import feature_selection
from sklearn import mixture
from sklearn import multiclass
from sklearn import multioutput
from sklearn import ensemble.partial_dependence
from sklearn import utils
from sklearn import sklearn.exceptions
from sklearn import sklearn.utils.sparsefuncs
from sklearn import sklearn.utils.fixes
from sklearn import sklearn.utils.multiclass
from sklearn import sklearn.utils.extmath
from sklearn import sklearn.utils.sparsetools
from sklearn import sklearn.utils.arrayfuncs
from sklearn import sklearn.utils.validation
from sklearn import sklearn.utils.graph_shortest_path
from sklearn import sklearn.utils.random
from sklearn import sklearn.utils.mask
from sklearn import sklearn.utils.lazy_urllib
from sklearn import sklearn.utils.bench
from sklearn import sklearn.utils.testing
from sklearn import sklearn.utils.mocking
from sklearn import sklearn.utils.deprecation
from sklearn import sklearn.utils._pprint
from sklearn import sklearn.utils._cx_oracle
from sklearn import sklearn.utils._logistic_sigmoid
from sklearn import sklearn.utils._blas
from sklearn import sklearn.utils._scipy_sparse_lsqr_backport
from sklearn import sklearn.utils._joblib
from sklearn import sklearn.utils._pilutil
from sklearn import sklearn.utils._scipy_linalg
from sklearn import sklearn.utils._scipy_sparse_csgraph
from sklearn import sklearn.utils._scipy_sparse_csgraph_shortest_path
from sklearn import sklearn.utils._scipy_sparse_csgraph_tests
from sklearn import sklearn.utils._scipy_sparse_csgraph_tests.test_shortest_path
from sklearn import sklearn.utils._scipy_sparse_csgraph_tests.test_traversal
from sklearn import sklearn.utils._scipy_sparse_csgraph_tests.test_connected_components
from sklearn import sklearn.utils._scipy_sparse_csgraph_tests.test_tools
from sklearn import sklearn.utils._scipy_sparse_csgraph_tests.test_matrix_validation
from sklearn import sklearn.utils._scipy_sparse_csgraph_tests.test_graph_utils
from sklearn import sklearn.utils._scipy_sparse_csgraph_tests.test_graph_laplacian
from sklearn import sklearn.utils._scipy_sparse_csgraph_tests.test_graph_shortest_path
from sklearn import sklearn.utils._scipy_sparse_csgraph_tests.test_graph_traversal
from sklearn import sklearn.utils._scipy_sparse_csgraph_tests.test_graph_connected_components
from sklearn import sklearn.utils._scipy_sparse_csgraph_tests.test_graph_tools
from sklearn import sklearn.utils._scipy_sparse_csgraph_tests.test_graph
```

9	203.80.214	97.210.52.21	1.2400	315800	1618400	TCP

In [101]:
```
fig = go.Figure()

fig.add_trace(go.Scattergeo:
    lon = top_10_pairs_location['lon'].append(top_10_pairs_location['lon2'],
    lat = top_10_pairs_location['lat'].append(top_10_pairs_location['lat2'],
    marker = dict:
        size = 2,
        color = 'rgb(255, 0, 0)',
        line = dict:
            width = 3,
            color = 'rgba(68, 68, 68, 0)'
    )
)

lons = []
lats = []
import numpy as np
lons = np.empty(2 * top_10_pairs_location.shape[0])
lats = np.empty(2 * top_10_pairs_location.shape[0])
lons[::2] = top_10_pairs_location['lon']
lats[::2] = top_10_pairs_location['lat']
lons[:1:2] = None
lons[:1:2] = top_10_pairs_location.shape[0]
lats[::2] = top_10_pairs_location['lat2']
lats[:1:2] = top_10_pairs_location['lat2']
lats[:1:2] = None

fig.add_trace:
    go.Scattergeo:
        lon = lons,
        lat = lats,
        mode = 'lines',
        line = dict:width = 1,color = 'red',
        opacity = 1
)

fig.update_layout:
    showlegend = False,
    geo = go.layout.Geo:
        showland = True,
    ),
    height=700,
)

fig.update_layout:(
    geo:
        projection:: 50
    )
)

fig.show:
```

As you can see, the top 10 pairs mainly connect Singapore with USA and UK

Top 5% Network Graph

In this part, we will inspect the top 5% of the interaction of the talker-listener directional pairs based on the number of packets send between the pair in the given sample of data.

In [102]:
```
t_1_pairs = data.groupby:['src_IP','dst_IP'].size:).reset_index(name:' # of Packets')
t_1_pairs = t_1_pairs.sort_values:[' # of Packets'], ascending=False:
t_1_pairs
```

Out[102]:

	src_IP	dst_IP	# of Packets
4236	193.62.192.8	137.132.228.15	3041
975	130.14.250.11	103.37.198.100	2599
1607	14.139.196.94	192.101.107.153	2368
1639	140.112.8.139	103.21.126.2	2056
1297	137.132.228.15	193.62.192.8	1910
...	...	...	...
2748	155.69.193.14	172.217.27.1	1
2747	155.69.193.108	172.217.27.3	1
2745	155.69.193.115	74.125.130.188	1
2744	155.69.193.112	74.125.68.99	1
6588	fe80:0000:0000:0000:b2a5:8a51:ca76:8776	fe80:0000:0000:0000:c4e:24ff:fe6c:8314	1

6589 rows × 3 columns

In [103]:
```
persons_5 = int(t_1_pairs.shape[0]/20)
```

In [104]:
```
top_5p_pairs = t_1_pairs.sort_values:[' # of Packets'], ascending=False:
top_5p_pairs = top_5p_pairs.head(persons_5):
top_5p_pairs
```

Out[104]:

	src_IP	dst_IP	# of Packets
4236	193.62.192.8	137.132.228.15	3041
975	130.14.250.11	103.37.198.100	2599
1607	14.139.196.94	192.101.107.153	2368
1639	140.112.8.139	103.21.126.2	2056
1297	137.132.228.15	193.62.192.8	1910
...	...	...	...
4250	185.231.231.157	140.110.112.99	26
1853	140.116.168.246	203.80.21.41	26
1057	130.246.176.41	146.115.32.42	26
222	193.6.192.11	140.111.1.2	26
1712	137.132.232.202	158.246.2.202	26

329 rows × 3 columns

In [105]:
```
pairs_5 = [tuple(row) for row in top_5p_pairs.to_numpy()]
len(top_5p_pairs)
```

Out[105]:
```
329
```

In [106]:
```
network_graph = igraph.Graph.TupleList:
    pairs_5,
    directed=True,
    weights=True,
)

network_graph.vs['size'] = 15
network_graph.vs['label_size'] = 0
network_graph.vs['label'] = network_graph.vs['name']
network_graph.vs['color'] = 'powderblue'

network_graph.es['width'] = (
    maximum.log(weight, 5.1)
    for weight in network_graph.es['weight']
)
network_graph.es['arrow_size'] = 1

network_graph.es['color'] = 'dodgerblue'


layout = network_graph.layout:'fr':
igraph.plot(network_graph, layout=layout, bbox=(2560, 1440)):
igraph.plot(network_graph, 'computer_network5.png', layout=layout, bbox=(2560, 1440)):
Image('computer_network5.png')
```

Out[106]:

In the graph above each node represents each IP address of the network nodes in the top 5% of the sample data. The directed edges represent packets' paths.

Top 1% Network Graph

In this part, we will inspect the top 1% of the interaction of the talker-listener directional pairs based on the number of packets send between the pair in the given sample of data.

In [107]:
```
persons_1 = int(t_1_pairs.shape[0]/100)
```

In [108]:
```
top_1p_pairs = t_1_pairs.sort_values:[' # of Packets'], ascending=False:
top_1p_pairs = top_1p_pairs.head(persons_1):
top_1p_pairs
```

Out[108]:

	src_IP	dst_IP	# of Packets
4236	193.62.192.8	137.132.228.15	3041
975	130.14.250.11	103.37.198.100	2599
1607	14.139.196.94	192.101.107.153	2368
1639	140.112.8.139	103.21.126.2	2056
1297	137.132.228.15	193.62.192.8	1910
...	...	...	...
1667	140.138.144.170	192.248.87.225	186
4236	194.68.32.16	202.158.207.20	186
4213	193.49.97.157	137.132.2.10	183
1172	134.219.229.225	140.113.55.80	179
1108	133.39.224.12	137.132.243.142	177

65 rows × 3 columns

In [109]:
```
pairs_1 = [tuple(row) for row in top_1p_pairs.to_numpy()]
len(top_1p_pairs)
```

Out[109]:
```
65
```

In [110]:
```
network_graph = igraph.Graph.TupleList:
    pairs_1,
    directed=True,
    weights=True,
)

network_graph.vs['size'] = 15
network_graph.vs['label_size'] = 0
network_graph.vs['label'] = network_graph.vs['name']
network_graph.vs['color'] = 'powderblue'

network_graph.es['width'] = (
    maximum.log(weight, 5.1)
    for weight in network_graph.es['weight']
)
network_graph.es['arrow_size'] = 1

network_graph.es['color'] = 'dodgerblue'


# plot graph

layout = network_graph.layout:'fr':
igraph.plot(network_graph, layout=layout, bbox=(2560, 1440)):
igraph.plot(network_graph, 'computer_network1.png', layout=layout, bbox=(2560, 1440)):
Image('computer_network1.png')
```

Out[110]:

The above graph represents the interactions between the nodes in the top 1% of the most popular talker-listener pairs.

Port Scanning

This part will focus on whether a node is being port scanned by another node.

In [111]:
```
port_scanning = pd.DataFrame:data.groupby:['src_IP','dst_IP'].['dst_port'].nunique():
port_scanning = port_scanning.reset_index():
port_scanning.sort_values:['port_scanned'], ascending=False:.head(20)
```

Out[111]:

	src_IP	dst_IP	port_scanned
4	193.62.192.8	137.132.228.15	1447
137.132.228.15	**193.62.192.8**	**1009**	
0	**0**	**90**	
59.191.192.7	**150.164.15.52**	**90**	
203.117.152.208	**103.37.198.23**	**81**	
76.125.10.59	**133.130.64.29**	**60**	
166.71.44.58	**192.3.219.10**	**57**	
207.241.226.123	**163.16.8.34**	**56**	
140.90.101.61	**167.205.32.8**	**51**	
192.3.219.10	**166.71.44.58**	**50**	
167.205.32.8	**140.90.101.61**	**50**	
202.21.158.254	**175.106.15.109**	**53**	
4	**0**	**45**	
204.60.254	**1**	**45**	
202.21.158.254	**134.197.62.20**	**42**	
166.71.44.163	**216.165.163.1**	**39**	
13.107.4.50	**202.21.158.37**	**36**	
31.13.76.52	**202.21.158.241**	**36**	
143.117.65.1	**163.16.8.34**	**36**	
184.224.137.58	**165.237.142.250**	**31**	

We group the entries by their source and destination IP address. In this way, we can create a new column to to count the unique destination ports. This will indicate if a talker communicates with many ports of a listener.

In [112]:
```
sus_data = data[(data['src_IP'] == '193.62.192.8') & (data['dst_IP'] == '137.132.228.15')] [data['src_IP'] == '137.132.228.15'] & (data['dst_IP'] == '193.62.192.8')]
sus_data['src_IP', 'dst_IP', 'dst_port']
```

Out[112]:

	src_IP	dst_IP	dst_port
4	137.132.228.15	193.62.192.8	34282
9	193.62.192.8	137.132.228.15	53923
12	193.62.192.8	137.132.228.15	53923
19	137.132.228.15	193.62.192.8	20480
38	193.62.192.8	137.132.228.15	53923
...	...	...	...
69285	193.62.192.8	137.132.228.15	34753
69501	137.132.228.15	193.62.192.8	3123
69536	137.132.228.15	193.62.192.8	49413
69542	193.62.192.8	137.132.228.15	57296
69567	137.132.228.15	193.62.192.8	3123

4951 rows × 3 columns

In [113]:
```
scanner_data = pd.DataFrame:
scanner_data['connection'] = sus_data['src_IP'] + " - " + sus_data['dst_IP']
scanner_data['dst_port'] = sus_data['dst_port']
scanner_data
```

Out[113]:

	connection	dst_port
4	137.132.228.15 - 193.62.192.8	34282
9	193.62.192.8 - 137.132.228.15	53923
12	193.62.192.8 - 137.132.228.15	53923
19	137.132.228.15 - 193.62.192.8	20480
38	193.62.192.8 - 137.132.228.15	53923
...	...	...
69285	193.62.192.8 - 137.132.228.15	34753
69501	137.132.228.15 - 193.62.192.8	3123
69536	137.132.228.15 - 193.62.192.8	49413
69542	193.62.192.8 - 137.132.228.15	57296
69567	137.132.228.15 - 193.62.192.8	3123

4951 rows × 2 columns

In [115]:
```
plt.figure:figsize=(8,10):
sb.stripplot:data=scanner_data, x='connection', y='dst_port'
```

Out[115]:
```
<AxesSubplot:xlabel='connection', ylabel='dst_port'>
```