MASTER THESIS FOR THE PPS (MS) IN ELECTRONIC PHYSICS (RADIOELECTROLOGY)

A SUPERVISED MACHINE LEARNING FRAMEWORK FOR ANOMALY-BASED INTRUSION DETECTION

Anastasia Chaloulakou (10024)

SUPERVISORS: K. SIOZIOS, M. DASYGENIS

Outline

- Introduction – general idea
- Pre-processing of NSL-KDD dataset
  - *Dataset features*
  - *Pre-processing steps*
- Evaluation and results
  - *Machine learning methods developed*
  - *Comparison of performances*
- Comparison to state-of-the-art and discussion

INTRODUCTION

Anomaly-based intrusion detection

- Cyber security is challenged:
  - *Vast amount of network traffic*
  - *Evolution and sophistication of malicious activities*
  - *Signature-based IDS can’t keep up with the new attacks increasing rate*

- Anomaly-based Intrusion Detection Systems
  - *Rely heavily on machine learning*
  - *Classify data based on normal or deviant behaviour*
  - *Anomalies can be caused by malicious actors, or performance-related*
- Solution → machine learning classification

Machine Learning methods

- Supervised:
  - *Best accuracy*
  - *Feature selection*
  - *Most reliable*
  - *Measurable performance*
- Problems:
  - *Label creation*
  - *Balanced representation of all classes*

- Semi-supervised:
  - *Use mechanisms like AE*
  - *Use labelled and/or unlabelled training data*
- Problems:
  - *Not good enough performance and no validation*
  - *Complexity of algorithms*

- Unsupervised:
  - *Newest methods*
  - *Can use real traffic for training (unlabelled data)*
- Problems:
  - *Not good enough performance and no validation*
  - *Complexity of algorithms*

Challenges of network security and anomaly detection

- Rapid development of networks and attacks today
- Reliance of our society on the Internet (more data generated every year)
- IoT, lower level devices connected to each other handling sensitive data

- Unavailability of open network datasets, especially recent ones
- Incompetence of unsupervised learning methods

Objectives

- Analyse the NSL-KDD dataset as a benchmark dataset for intrusion detection
- Compare three scenarios:
  - *Multiclass classification (40 labels) – for each different attack*
  - *Grouped classification (5 labels) – attacks are grouped together by kind*
  - *Binary classification (2 labels) – normal and abnormal traffic classification*
- Pre-process the dataset so that it is usable by the models
- Develop and compare 5 common supervised machine learning classification algorithms
- Evaluate results and compare to research

PRE-PROCESSING OF THE NSL-KDD DATASET

Why the NSL-KDD

- A dataset of network traffic records
- Created in 2009, curated from KDDCup99 (1999)
- Pros:
  - *One of the few publicly available datasets*
  - *Already labelled (necessary for supervised learning)*
  - *Rich in features*
  - *Still used in research*
- Cons:
  - *Outdated (doesn’t contain recent attacks)*
  - *Synthetic dataset*

Characteristics

- Consists of .csv files:
  - $KDDTrain+$ *with* $125{,}973$ *data entries*
  - $KDDTest+$ *with* $22{,}544$ *data entries* $\rightarrow$ $(17.9\%\ rate)$
- Difficulty levels: 21
  - 49.66% *of training set and* 47.44% *of test set are 21/21 level*
- 43 columns: 1-41 → features, 42 → label, 43 → difficulty level
- Subsets: *KDDTest-21, KDDTraini+_20Percent*
  - *Their records are all included in the bigger datasets*

Labels (traffic type)

Class	R2L	DoS	U2R	Probe
Attacks	ftp_write	apache2	buffer_overflow	ipsweep
	guess_passwd	back	loadmodule	mscan
	httptunnel	land	perl	nmap
	imap	neptune	ps	portsweep
	multihop	mailbomb	rootkit	saint
	named	pod	sqlattack	satan
	phf	processtable	xterm	
	sendmail	smurf		
	snmpgetattack	teardrop		
	spy	udpstorm		
	snmpguess	worm		
	warezmaster			
	warezclient			
	xlock			
	xsnoop			
Total	15	11	7	6

- 39 attacks + normal traffic
- Groups of attacks:
  - *Denial of Service (DoS)*
  - *Remote to Local (R2L)*
  - *User to Root (U2R)*
  - *Probe*

Distribution of labels by class and subset

Type of traffic	# in training set	% in training set	# in test set	% in test set
normal	67343	53.46%	9711	43.08%
DoS	45927	36.46%	7460	33.09%
Probe	11656	9.25%	2885	12.79%
R2L	995	0.79%	2421	10.74%
U2R	52	0.04%	67	0.30%

- Skewed (but realistic) distribution towards normal and DoS traffic
- Differences:
  - *In test set, normal traffic is not more than half of the total*
  - *Boost in R2L attacks*
  - *In training set there are 23 different labels, in test set there are 38 labels*
- Important to test the model with attacks not encountered during training

Features

- Col.1: Duration of connection
- Col.2: Protocol (TCP, UDP, ICMP)
- Col.3: Services (http, DNS request, email…)
- Col. 4: Flags
- Col.5-9: Header info
- Col. 10-22: Connection-based info (from payload)
- Col. 23-31: Time-based info (traffic analysed over a 2 sec. window)
- Col. 32-42: Host-based info (over multiple connections)

Pre-processing Steps

- **DATAFRAME CREATION**
  - Import .csv files
  - Create separate instances (multiclass, binary, grouped)
- **DATA CLEANING**
  - Clean out empty/wrong cells
  - Separate last column
- **ONE-HOT ENCODING**
  - Categorical values need to be numerical
- **X AND Y COMPONENTS**
  - Separate features from labels
- **ALIGNMENT**
  - Training and test sets need to be uniform
- **SCALING**
  - Minimize bias
  - All variables in 0-1 range
- **INPUT TO MODELS**

Dataframes creation

- Import the .csv files $KDDTrain+$ and $KDDTest+$ *(pandas library)*
  - *Dataframe type variables with sizes* $125{,}973 \times 43$ *and* $22{,}544 \times 43$ *respectively*

- Create 2 more copies and format the labels
  - *Binary: rename all attacks, so that there are only "normal" and "abnormal" labels*
  - *4-class grouping: rename attacks according to their attack class, labels are "normal", "DoS", "Probe", "R2L" and "U2R"*

```
df_nsltrain:
   0    1         2   3    4     5  6  7  8  9  ...    33    34    35  \
0  0  tcp  ftp_data  SF  491     0  0  0  0  0  ...  0.17  0.03  0.17
1  0  udp     other  SF  146     0  0  0  0  0  ...  0.00  0.60  0.88
2  0  tcp   private  S0    0     0  0  0  0  0  ...  0.10  0.05  0.00
3  0  tcp      http  SF  232  8153  0  0  0  0  ...  1.00  0.00  0.03
4  0  tcp      http  SF  199   420  0  0  0  0  ...  1.00  0.00  0.00

     36    37    38    39    40       41  42
0  0.00  0.00  0.00  0.05  0.00   normal  20
1  0.00  0.00  0.00  0.00  0.00   normal  15
2  0.00  1.00  1.00  0.00  0.00  neptune  19
3  0.04  0.03  0.01  0.00  0.01   normal  21
4  0.00  0.00  0.00  0.00  0.00   normal  21
```

```
df_nsltrain_4clas:
   0    1         2   3    4     5  6  7  8  9  ...    33    34    35  \
0  0  tcp  ftp_data  SF  491     0  0  0  0  0  ...  0.17  0.03  0.17
1  0  udp     other  SF  146     0  0  0  0  0  ...  0.00  0.60  0.88
2  0  tcp   private  S0    0     0  0  0  0  0  ...  0.10  0.05  0.00
3  0  tcp      http  SF  232  8153  0  0  0  0  ...  1.00  0.00  0.03
4  0  tcp      http  SF  199   420  0  0  0  0  ...  1.00  0.00  0.00

     36    37    38    39    40      41  42
0  0.00  0.00  0.00  0.05  0.00  normal  20
1  0.00  0.00  0.00  0.00  0.00  normal  15
2  0.00  1.00  1.00  0.00  0.00     DoS  19
3  0.04  0.03  0.01  0.00  0.01  normal  21
4  0.00  0.00  0.00  0.00  0.00  normal  21
```

Data Cleaning

- Confirm that there are no missing, wrong format, out-of-bounds and redundant values
  - *All records are unique and with all features*

- Separate last column (difficulty level)
  - *No real information for the model, only for us to compare training and test set*

- Drop col. 20 (number of outbound commands in an ftp session)
  - *All $0s$, became NaN during correlation calculations*

One-hot Encoding

- Turn the categorical values into numerical
  - *Counted in correlation calculations*
  - *Compatible for the model*
- Create *dummy* variables: one label is turned into a N*-dimensional vector*
  - *N is the number of all different values the categorical variable has*

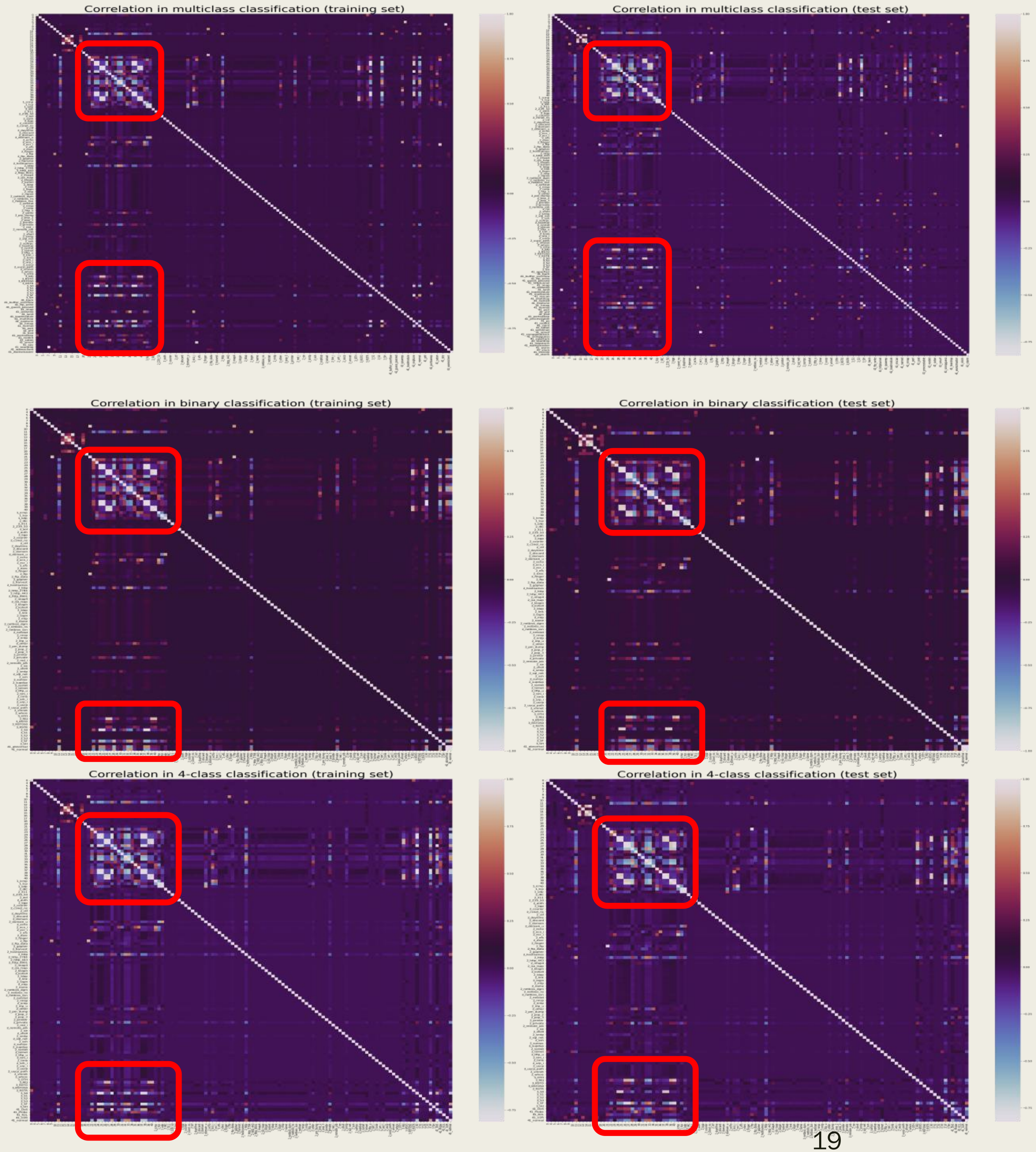
    e.g. Column 2: $\{TCP, UDP, ICMP\} \rightarrow \{[1,0,0], [0,1,0], [0,0,1]\}$
  - *Each record has all* $0s$*, except in one dimension that it has* 1
  - *.get_dummies method: move categorical values at the end and expand them into vectors*

Labels of columns before	Labels of columns in multiclass training dataframe, after one-hot encoding	Labels of columns in binary classification training dataframe, after one-hot encoding	Labels of columns in 4-class classification training dataframe, after one-hot encoding
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41]	[0, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, '1_icmp', '1_tcp', '1_udp', '2_IRC', '2_X11', '2_Z39_50', '2_aol', '2_auth', '2_bgp', '2_courier', '2_csnet_ns', '2_ctf', '2_daytime', '2_discard', '2_domain', '2_domain_u', '2_echo', '2_eco_i', '2_ecr_i', '2_efs', '2_exec', '2_finger', '2_ftp', '2_ftp_data', '2_gopher', '2_harvest', '2_hostnames', '2_http', '2_http_2784', '2_http_443', '2_http_8001', '2_imap4', '2_iso_tsap', '2_klogin', '2_kshell', '2_ldap', '2_link', '2_login', '2_mtp', '2_name', '2_netbios_dgm', '2_netbios_ns', '2_netbios_ssn', '2_netstat', '2_nnsp', '2_nntp', '2_ntp_u', '2_other', '2_pm_dump', '2_pop_2', '2_pop_3', '2_printer', '2_private', '2_red_i', '2_remote_job', '2_rje', '2_shell', '2_smtp', '2_sql_net', '2_ssh', '2_sunrpc', '2_supdup', '2_systat', '2_telnet', '2_tftp_u', '2_tim_i', '2_time', '2_urh_i', '2_urp_i', '2_uucp', '2_uucp_path', '2_vmnet', '2_whois', '3_OTH', '3_REJ', '3_RSTO', '3_RSTOS0', '3_RSTR', '3_S0', '3_S1', '3_S2', '3_S3', '3_SF', '3_SH', '41_back', '41_buffer_overflow', '41_ftp_write', '41_guess_passwd', '41_imap', '41_ipsweep', '41_land', '41_loadmodule', '41_multihop', '41_neptune', '41_nmap', '41_normal', '41_perl', '41_phf', '41_pod', '41_portsweep', '41_rootkit', '41_satan', '41_smurf', '41_spy', '41_teardrop', '41_warezclient', '41_warezmaster']	[0, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, '1_icmp', '1_tcp', '1_udp', '2_IRC', '2_X11', '2_Z39_50', '2_aol', '2_auth', '2_bgp', '2_courier', '2_csnet_ns', '2_ctf', '2_daytime', '2_discard', '2_domain', '2_domain_u', '2_echo', '2_eco_i', '2_ecr_i', '2_efs', '2_exec', '2_finger', '2_ftp', '2_ftp_data', '2_gopher', '2_harvest', '2_hostnames', '2_http', '2_http_2784', '2_http_443', '2_http_8001', '2_imap4', '2_iso_tsap', '2_klogin', '2_kshell', '2_ldap', '2_link', '2_login', '2_mtp', '2_name', '2_netbios_dgm', '2_netbios_ns', '2_netbios_ssn', '2_netstat', '2_nnsp', '2_nntp', '2_ntp_u', '2_other', '2_pm_dump', '2_pop_2', '2_pop_3', '2_printer', '2_private', '2_red_i', '2_remote_job', '2_rje', '2_shell', '2_smtp', '2_sql_net', '2_ssh', '2_sunrpc', '2_supdup', '2_systat', '2_telnet', '2_tftp_u', '2_tim_i', '2_time', '2_urh_i', '2_urp_i', '2_uucp', '2_uucp_path', '2_vmnet', '2_whois', '3_OTH', '3_REJ', '3_RSTO', '3_RSTOS0', '3_RSTR', '3_S0', '3_S1', '3_S2', '3_S3', '3_SF', '3_SH', '41_abnormal', '41_normal']	[0, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, '1_icmp', '1_tcp', '1_udp', '2_IRC', '2_X11', '2_Z39_50', '2_aol', '2_auth', '2_bgp', '2_courier', '2_csnet_ns', '2_ctf', '2_daytime', '2_discard', '2_domain', '2_domain_u', '2_echo', '2_eco_i', '2_ecr_i', '2_efs', '2_exec', '2_finger', '2_ftp', '2_ftp_data', '2_gopher', '2_harvest', '2_hostnames', '2_http', '2_http_2784', '2_http_443', '2_http_8001', '2_imap4', '2_iso_tsap', '2_klogin', '2_kshell', '2_ldap', '2_link', '2_login', '2_mtp', '2_name', '2_netbios_dgm', '2_netbios_ns', '2_netbios_ssn', '2_netstat', '2_nnsp', '2_nntp', '2_ntp_u', '2_other', '2_pm_dump', '2_pop_2', '2_pop_3', '2_printer', '2_private', '2_red_i', '2_remote_job', '2_rje', '2_shell', '2_smtp', '2_sql_net', '2_ssh', '2_sunrpc', '2_supdup', '2_systat', '2_telnet', '2_tftp_u', '2_tim_i', '2_time', '2_urh_i', '2_urp_i', '2_uucp', '2_uucp_path', '2_vmnet', '2_whois', '3_OTH', '3_REJ', '3_RSTO', '3_RSTOS0', '3_RSTR', '3_S0', '3_S1', '3_S2', '3_S3', '3_SF', '3_SH', '41_DoS', '41_Probe', '41_R2L', '41_U2R', '41_normal']
In total: 42	In total: 144	In total: 123	In total: 126

Correlation

- Find the pair-wise relationships between all (numerical) features
  - *Using the .corr() pandas function*

- Range: $[-1, +1]$:
  - $c \rightarrow -1$*: inversely proportional values*
  - $c \rightarrow 1$*: proportional values*
  - $c \rightarrow 0$*: irrelevant values*



X and Y components – Alignment Standard scaling

- Split the dataframes into features X (col. 1- 41) and labels Y (col. 42)
  - *One-hot encode the* X *component, leave* Y *as labels (output)*
- *Alignment:* training X component is 125973×121, test X is 22544×115
  - *Length difference doesn't matter, but features dimensions need to be the same*
  - *Fill the empty values from extra columns with 0 in the right place*
- Standard scaler: $x' = \frac{x-\mu}{s}$ *(normal distribution)*

  x'*: new scaled value,* x*: original data value,* μ*: mean of training samples,* s*: standard deviation*
- Two steps:
  - *Fitting: computes mean and standard deviation of the data → training set only*
  - *Transformation: perform the scaling on the data → both sets*

EVALUATION AND RESULTS

Machine learning classification models

- Logistic Regression
  - *Use the sigmoid function to test each class at a time*
- Decision Tree
  - *Create subsets (classes) based on questions posed on the dataset*
- K – Nearest Neighbours
  - *Classify with no hypotheses or conditions*
- Gaussian Naïve Bayes
  - *Conditional probability model based on Bayes theorem*
- Multi – Layer Perceptron
  - *Basic ANN architecture*

Deviant performances

- Training and test sets are very different
  - *Distribution and types of labels*
  - *Difficulty levels*
  - *Services and flags (highly correlated with many other features and with output)*

- Check for overfitting:
  - *Case A: training set* $KDDTrain+$ *and test set (validation)* $KDDTest+$
  - *Case B: training and test set are part of* $KDDTrain+$ *(using .train_test_split)*

Accuracy

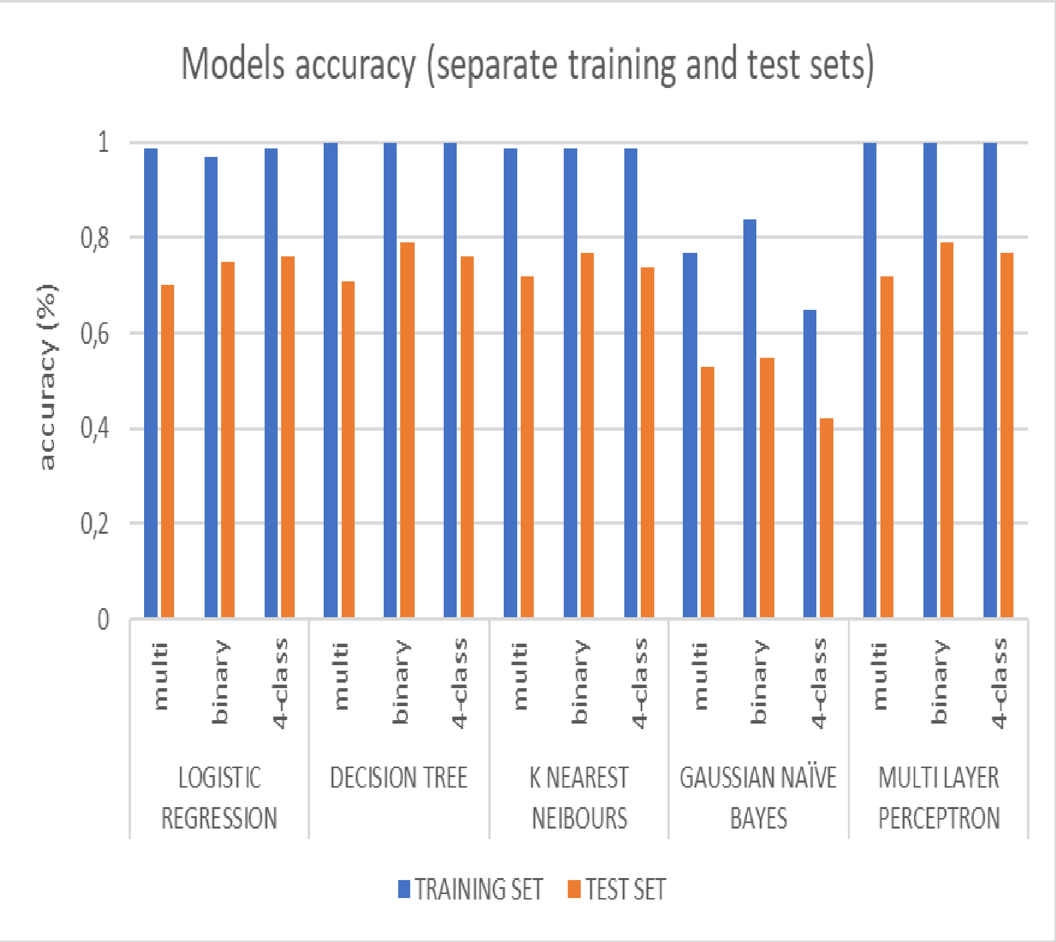
- Overall ability of the model to classify correctly over all of the values

- $accuracy\left(y, \widehat{y}\right) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1\left(\widehat{y_i} = y_i\right)$
  - $\hat{y}_i$: *predicted output*, y_i: *real value*

- $accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{correct\ classifications}{all\ classifications}$

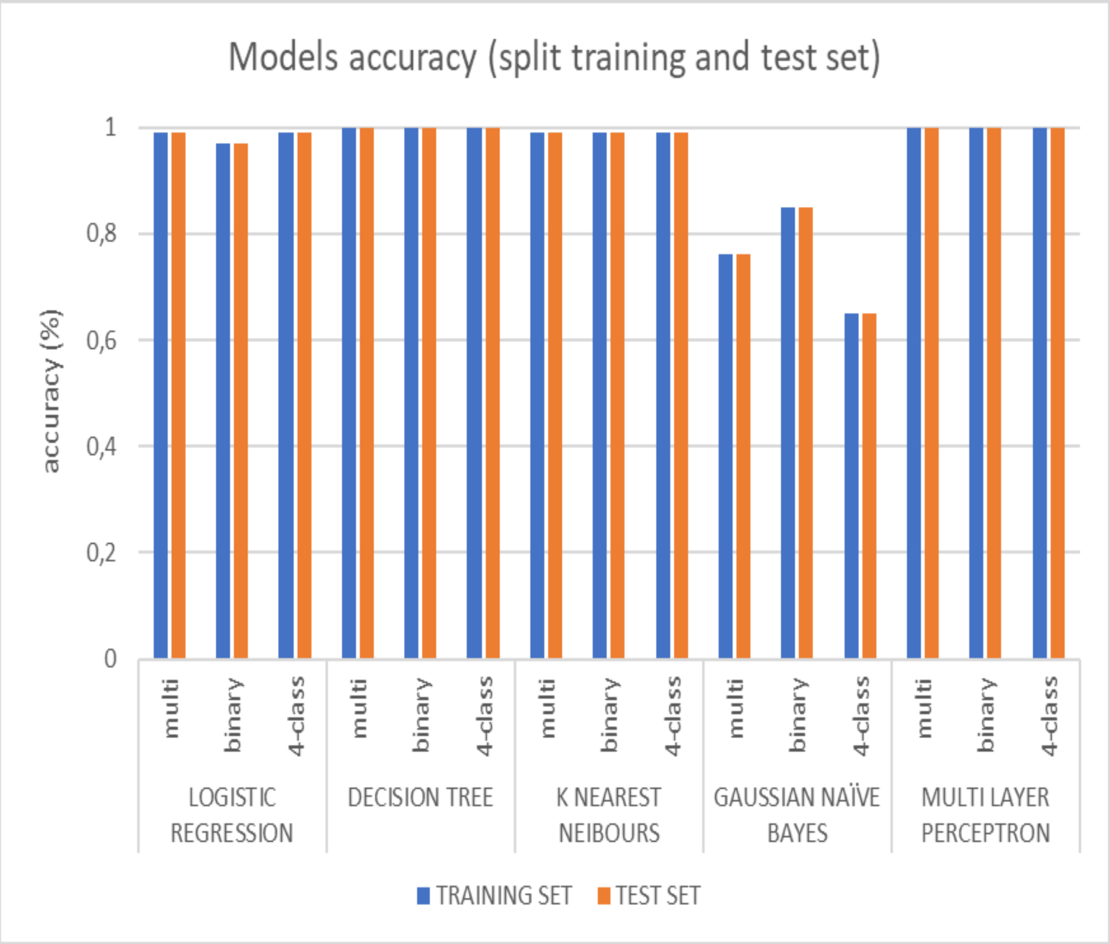
Case A (separate training and test sets)

	CLASSIFICATION ALGORITHM	CLASS SCENARIO	TRAINING SET	TEST SET
CASE A: using the KDDTrain+ and KDDTest+ as training and test sets	LOGISTIC REGRESSION	multi	0,99	0,70
		binary	0,97	0,75
		4-class	0,99	0,76
	DECISION TREE	multi	1,00	0,71
		binary	1,00	0,79
		4-class	1,00	0,76
	K NEAREST NEIBOURS	multi	0,99	0,72
		binary	0,99	0,77
		4-class	0,99	0,74
	GAUSSIAN NAÏVE BAYES	multi	0,77	0,53
		binary	0,84	0,55
		4-class	0,65	0,42
	MULTI LAYER PERCEPTRON	multi	1,00	0,72
		binary	1,00	0,79
		4-class	1,00	0,77



Case B (split training and validation set)

	CLASSIFICATION ALGORITHM	CLASS SCENARIO	TRAINING SET	TEST SET
CASE B: splitting the KDDTrain+ in training and test/validation subsets	LOGISTIC REGRESSION	multi	0,99	0,99
		binary	0,97	0,97
		4-class	0,99	0,99
	DECISION TREE	multi	1,00	1,00
		binary	1,00	1,00
		4-class	1,00	1,00
	K NEAREST NEIBOURS	multi	0,99	0,99
		binary	0,99	0,99
		4-class	0,99	0,99
	GAUSSIAN NAÏVE BAYES	multi	0,76	0,76
		binary	0,85	0,85
		4-class	0,65	0,65
	MULTI LAYER PERCEPTRON	multi	1,00	1,00
		binary	1,00	1,00
		4-class	1,00	1,00



Evaluation

- Case B is not valuable or useful
  - *Traffic data won't be so close to training data in real life*
  - *Most models use the same dataset split for training and validation*

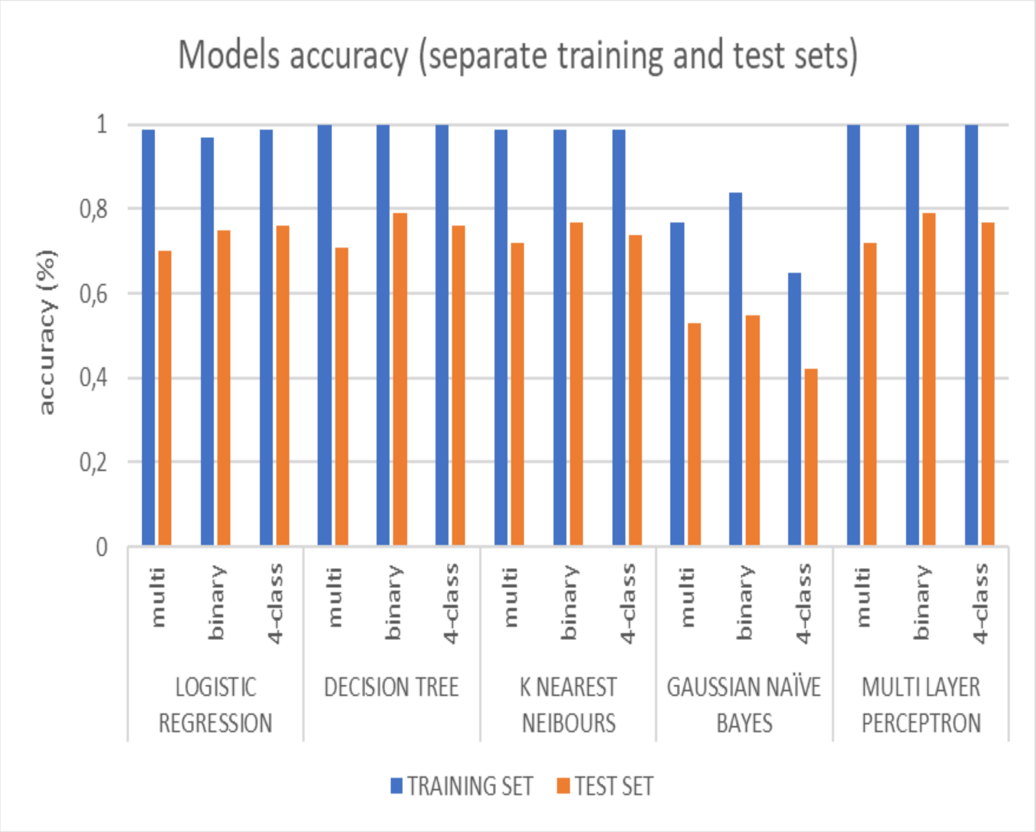
- Classification reports
  - *Detailed analysis of all models and classifications performance label by label*
  - *All metrics* $\rightarrow 1$ *for frequently encountered outputs and* $\rightarrow 0$ *for scarce outputs*

Evaluation of models

- Less classes ↔ higher accuracy

- Highest: DT+MLP binary classification (79%)
- Lowest: LR+DT multiclass classification (70%)

- GNB $(53\%, 55\%, 42\%)$:
  - *No Gaussian distribution of the data*
  - *Many highly correlated features*



Relevant research (2018-2022)

- [1], [2]: combined $KDDTrain+$ and $KDDTest+$ into one and split it *(similar to case B)*
  - accuracy $99-99{,}6\%$
  - DT, RF, MLP algorithms

- [3]: with PCA, reduced to 6 features
  - All features → accuracy $74-79\%$
  - Reduced → $71-75\%$
  - DT, DNN, PCA+DNN algorithms

Relevant research (2018-2022)

- [4]: input layer, multiple CNN, BLSTM, attention layer
  - Accuracy $84,2\%$
  - With DT, MLP, RF, accuracy $72 - 78\%$
- [5]: AE (115>50 features), sparse AE (50>10), LR (10>2)
  - Binary classification only
  - Accuracy $87,2\%$
- [6]: Best of performance of all with LSTM, DCNN, Denoising and Contractive AE
  - Accuracy $89\%\ (LSTM)$, $81 - 85\%\ (AE)$
  - RF, DR, k-NN, MLP algorithms → accuracy $74 - 82\%$

DISCUSSION – FUTURE WORK

This Project

- Basic (and outdated) algorithms gave results not far behind state-of-the-art
- Utilizes one of the most popular datasets available
- Compares five common classification methods
- Compares different classification scenarios
- Open to upgrades in both data and algorithms

- Can be comparable with recent research
  - Still use NSL-KDD as benchmark

Future work

- Use the NSL-KDD as an unsupervised dataset, and be able to validate results
  - *Attention mechanisms, AE, clustering methods*
- More advanced supervised methods (DNN, CNN, LSTM)

- Data-centric upgrade: use real data with unsupervised learning
  - *With data from a secure environment, AEs would perform very well*
  - *NSL-KDD can be used for validation of the unsupervised models*
  - *More advanced project: feature extraction/selection, data cleaning, unsupervised methods only*

Relevant research (2018-2022)

- [1] J. J. Estévez-Pereira, D. Fernández, and F. J. Novoa, “Network Anomaly Detection Using Machine Learning Techniques,” Aug. 2020, p. 8. doi: 10.3390/proceedings2020054008 (n.11)
- [2] O. Jamal Ibrahim *et al.*, “Network intrusion detection: a comparative study of four classifiers using the NSL-KDD and KDD’99 datasets,” *J. Phys*, p. 12043, 2022, doi: 10.1088/1742-6596/2161/1/012043. (n.13)
- [3] S. Rawat, A. Srinivasan, V. Ravi, and U. Ghosh, “Intrusion detection systems using classical machine learning techniques vs integrated unsupervised feature learning and deep neural network,” *Internet Technology Letters*, vol. 5, no. 1, Jan. 2022, doi: 10.1002/itl2.232. (n.9)
- [4] T. Su, H. Sun, J. Zhu, S. Wang, and Y. Li, “BAT: Deep Learning Methods on Network Intrusion Detection Using NSL-KDD Dataset,” *IEEE Access*, vol. 8, pp. 29575–29585, 2020, doi: 10.1109/ACCESS.2020.2972627. (n.36)
- [5] S. Gurung, M. K. Ghose, and A. Subedi, “Deep Learning Approach on Network Intrusion Detection System using NSL-KDD Dataset,” Computer Network and Information Security, vol. 3, pp. 8–14, 2019, doi: 10.5815/ijcnis.2019.03.02. (n.12)
- [6] S. Naseer *et al.*, “Enhanced network anomaly detection based on deep neural networks,” *IEEE Access*, vol. 6, pp. 48231–48246, Aug. 2018, doi: 10.1109/ACCESS.2018.2863036. (n.35)

Resources

- Python Jupyter notebook
- Libraries:
  - *Pandas*
  - *Numpy*
  - *Sklearn*
    - Linear_model, tree, neighbors, naive_bayes, neural_network
    - Preprocessing, metrics, model_selection
  - *Matplotlib*
  - *Seaborn*

Acknowledgments

I would like to thank my supervisors, mr. Kostas Siozios and mr. Minas Dasygenis, for their guidance and support throughout the whole process of creating this thesis.

I also want to thank Dimitris Tsiktsiris, for his thoughtful advice and help.

The IT department of AUTH, and especially mr. Georgios Pallas, for their help in gathering and preparing the traffic from the university gateway, even though that part of the project failed to launch in the end.

Lastly, I would like to express my gratitude to my family and friends, for the years of constant support, help and encouragement that they generously provided me with throughout my undergraduate and postgraduate studies, and for their patience while I was ranting about things they didn't understand.

This thesis was partially supported by the Hellenic Petroleum Company and the Special Account for Research Funds, through the Scholarship for Academic Excellence during the academic year 2020-2021

Thank you