INT246

CA-1

LOVELY PROFESSIONAL UNIVERSITY

Predicting Anomalies in Network Traffic [Virus(Malware) Identification using Deep Learning] –Project

Name of student: Putsala Sai Sri Ganesh
Registration Number: 11914316

Signature of the students: P.S.S.Ganesh

Contents

1 Introduction 3
1.1 Intrusion Detection Systems (IDS) 3
1.2 Problem Statement 3

2 Dataset Description and Exploration 3
2.1 Target Variable ”label” distribution 4
2.2 Categorical Columns Distributions 4
2.3 Continuous Columns 6

3 Data Normalization 8

4 Modeling 8
4.1 Model that over-fits the entire dataset 8
4.1.1 Model-1 Architecture Summary Diagram 8
4.1.2 Inference 8

5 Generalized Model building, selection and evaluation 8
5.1 Iteration-0: Baseline Modeling 9
5.2 Iteration-1: Modifying activation functions 9
5.3 Iteration-2: Modifying Optimizer functions 9
5.4 Iteration-3: Modifying Batch Size 9
5.5 Iteration-4: Modifying Number of Neurons 9
5.6 Iteration-5: Modifying Number of Epochs 10
5.6.1 Iteration-5 Model Performances 10
5.7 Model Selection 10
5.7.1 ROC Curves for all Models 10
5.8 Model Selection Inference 10

6 Overfitting with target variable 11

7 Custom predict function 11

8 Iterative Feature Reduction Selection 11
8.1 Feature Importance 11
8.2 Performance After Removal 14
8.3 Performance Comparison: All-Features-Model vs Important-Features-Model . . . 15

9 Conclusion 16

1 Introduction

1.1 Intrusion Detection Systems (IDS)

Intrusion Detection Systems (IDS) provide network security software applications by continuously monitoring network traffic and classifying the connections as normal or malicious. IDS are categorized into two types based on the responsive nature - Passive IDS and Active IDS. A passive IDS is designed to identify and block the malicious and malware attacks manually by human experts whereas active IDS is designed to identify and block the malware attacks using a software automatically.

IDS are also categorized into Signature based IDS and Anomaly Based IDS. In the Signature based IDS, there exists a database which contains details about all known malware attacks against which each network traffic connection is validated to identify the malicious nature if it exists. This type of IDS is costly and has to keep updating new types of attacks frequently.
Anomaly based IDS is a behavior-based system where any deviation from the normal network traffic patterns will be reported using pattern-recognition techniques. In this research paper, a proof-of-concept for a machine learning based Anomaly based intrusion detection will be evaluated on a benchmark intrusion detection dataset.

1.2 Problem Statement

In this research, KDD Cup 1999 dataset will be used to build a machine learning based intrusion detection problem to predict (classify) whether a network connection is ”normal” or ”abnormal”. As this is a categorical prediction, this is a binary classification problem

2 Dataset Description and Exploration

In this research, the KDD Cup 1999 dataset has been chosen for the data analysis and model building. This dataset was used in the fifth international conference for the data mining and knowledge discovery competition and contains a huge variety of network intrusion data that is simulated in an environment equipped with a military network setting.
The original source of the dataset is from the official KDD website [2] and can also be found in the kaggle website . The dataset contains about 494,000 records and 41 features (columns) which contains network traffic details like source bytes, destination connection information, type of attacks and so on. Below are the list of features (columns) and their data types in the dataset.

2.1 Target Variable ”label” distribution

The target variable ”label” contains all different types of malware attacks and also the value ”normal” (i.e., not an attack) as shown below.

’normal’, ’buffer overflow’, ’loadmodule’, ’perl’, ’neptune’, ’smurf’, ’guess passwd’, ’pod’, ’teardrop’, ’portsweep’, ’ipsweep’, ’land’, ’ftp write’, ’back’, ’imap’, ’satan’, ’phf’, ’nmap’, ’multihop’, ’warezmaster’, ’warezclient’, ’spy’, ’rootkit’.

All malware attacks are grouped (transformed) to the value ”abnormal” to make this problema binary classification problem instead of a multi-class classification. The target variable distribution can be found below (see **Figure 1**).

Figure 1: Target Variable ”label” Distribution.

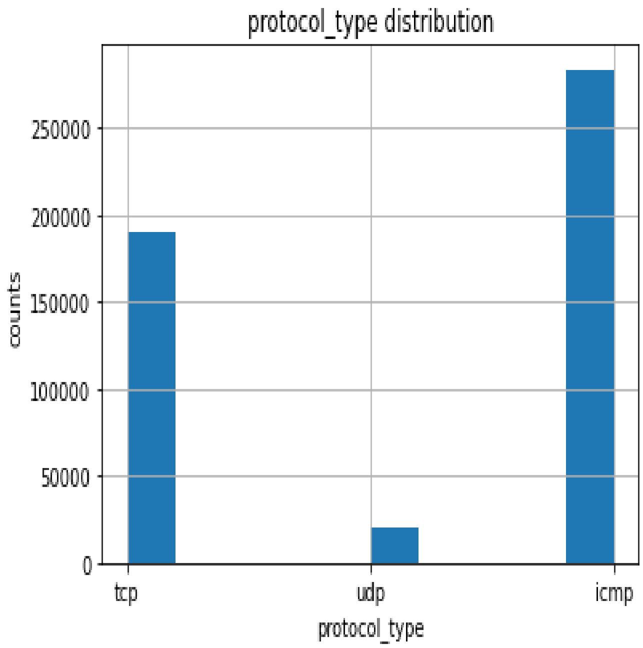
2.2 Categorical Columns Distributions

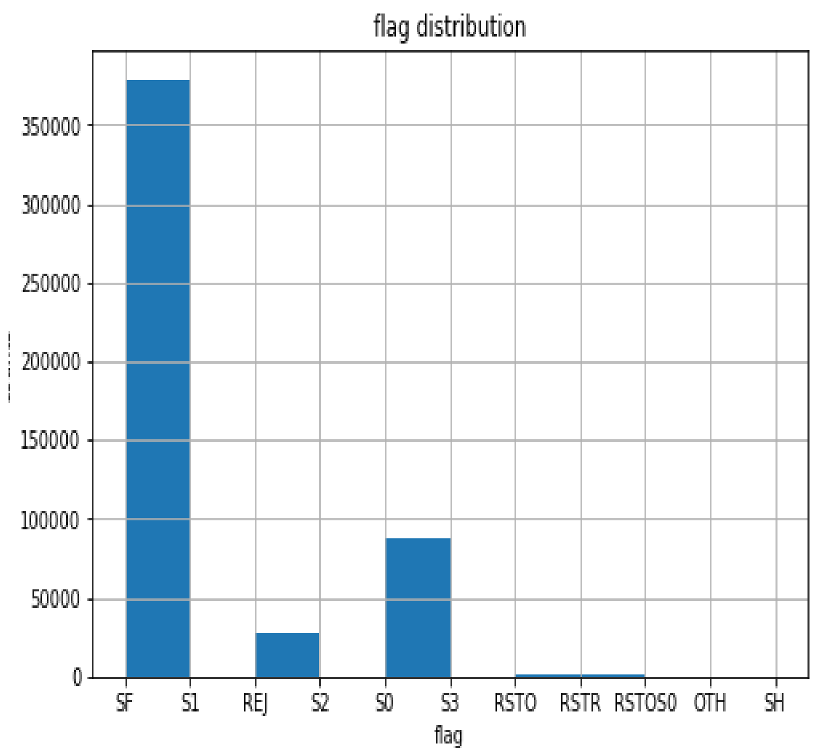
Below are the categorical columns in this dataset and their corresponding distribution plots. ’protocol type’, ’service’, ’flag’, ’land’, ’logged in’, ’is host login’, ’is guest login’

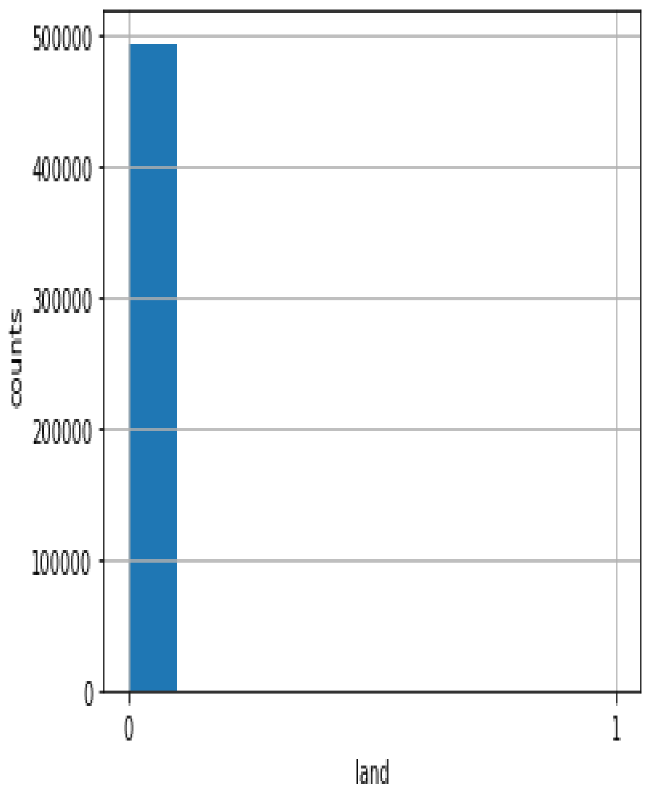
Categorical variables statistics can be found in the below table:

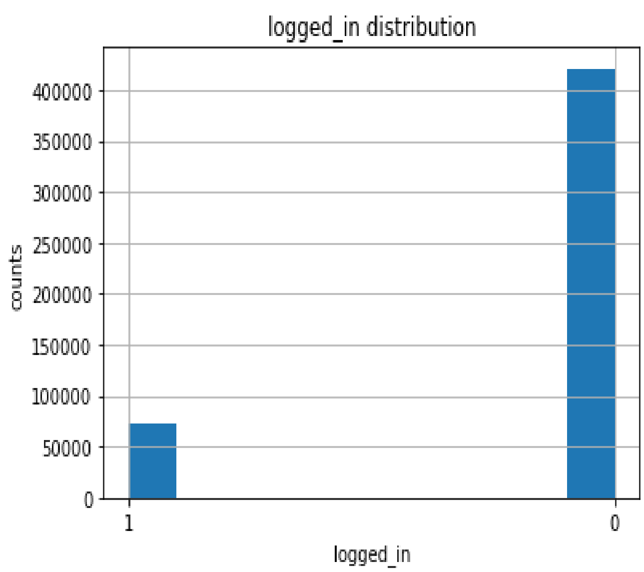
Table 1: Categorical Variables Statistics.

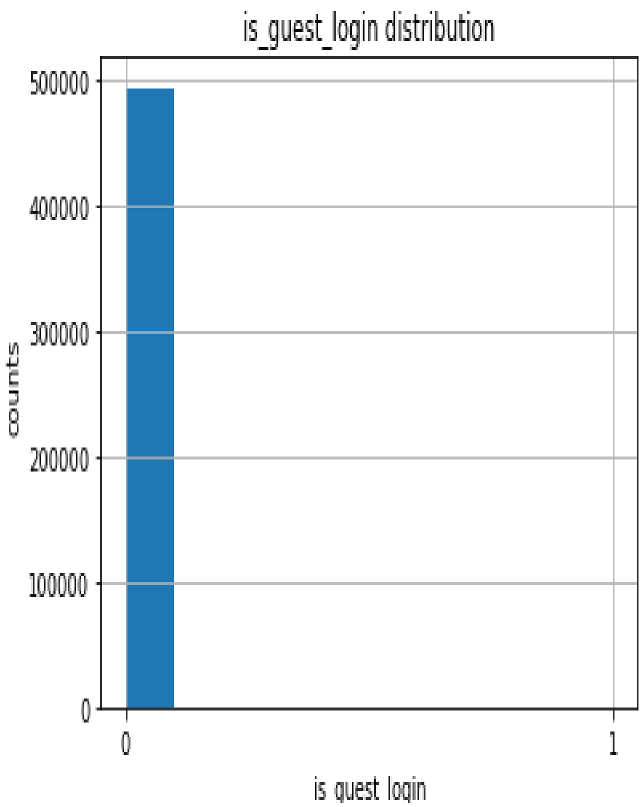
Stats	protocol_type	service	flag	land	logged_in	is_host_login	is_guest_login
count	494020	494020	494020	494020	494020	494020	494020
unique	3	66	11	2	2	1	2
top	icmp	ecr_i	SF	0	0	0	0
freq	283602	281400	378439	493998	420784	494020	493335

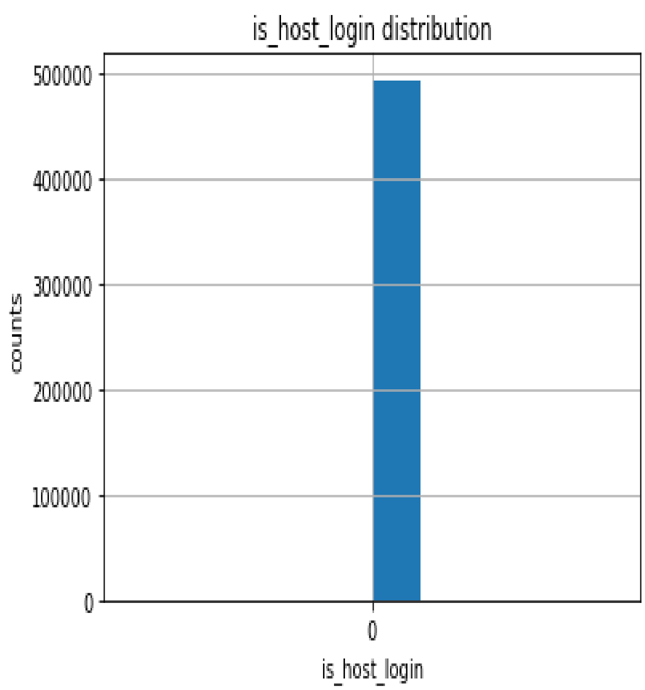

protocol_type distribution
counts
250000
200000
150000
100000
50000
0
tcp
udp
icmp
protocol_type


flag distribution
350000
300000
250000
200000
150000
100000
50000
0
SF
S1
REJ
S2
S0
S3
RSTO
RSTR
RSTOS0
OTH
SH
flag


500000
400000
300000
200000
100000
0
counts
0
1
land


logged_in distribution
400000
350000
300000
250000
200000
150000
100000
50000
0
counts
1
0
logged_in


is_guest_login distribution
500000
400000
300000
200000
100000
0
counts
0
1
is_guest_login


is_host_login distribution
500000
400000
300000
200000
100000
0
counts
0
is_host_login

2.3 Continuous Columns

Below are the continuous columns in this dataset and their corresponding distribution plots.

’duration’, ’src bytes’, ’dst bytes’, ’wrong fragment’, ’urgent’, ’hot’, ’num failed logins’, ’num compromised’, ’root shell’, ’su attempted’, ’num root’, ’num file creations’, ’num shells’, ’num access files’, ’num outbound cmds’, ’count’, ’srv count’, ’serror rate’, ’srv serror rate’, ’rerror rate’, ’srv rerror rate’, ’same srv rate’, ’diff srv rate’, ’srv diff host rate’, ’dst host count’, ’dst host srv count’, ’dst host same srv rate’, ’dst host diff srv rate’, ’dst host same src port rate’, ’dst host srv diff host rate’, ’dst host serror rate’, ’dst host srv serror rate’, ’dst host rerror rate’, ’dst host srv rerror rate’

Continuous variables statistics can be found in the below table:

Table 2: Categorical Variables Statistics.

Stats	count	mean	std	min	25%	50%	75%	max
duration	494020	47.9794	707.7472	0	0	0	0	58329
src_bytes	494020	3025.616	988219.1	0	45	520	1032	6.93E+08
dst_bytes	494020	868.5308	33040.03	0	0	0	0	5155468
wrong_fragment	494020	0.006433	0.134805	0	0	0	0	3
urgent	494020	1.42E-05	0.00551	0	0	0	0	3
hot	494020	0.034519	0.782103	0	0	0	0	30
num_failed_logins	494020	0.000152	0.01552	0	0	0	0	5
num_compromised	494020	0.010212	1.798328	0	0	0	0	884
root_shell	494020	0.000111	0.010551	0	0	0	0	1
su_attempted	494020	3.64E-05	0.007793	0	0	0	0	2
num_root	494020	0.011352	2.01272	0	0	0	0	993
num_file_creations	494020	0.001083	0.096416	0	0	0	0	28
num_shells	494020	0.000109	0.01102	0	0	0	0	2
num_access_files	494020	0.001008	0.036482	0	0	0	0	8
num_outbound_cmds	494020	0	0	0	0	0	0	0
count	494020	332.2864	213.1471	0	117	510	511	511
srv_count	494020	292.9071	246.3227	0	10	510	511	511
serror_rate	494020	0.176687	0.380717	0	0	0	0	1
srv_serror_rate	494020	0.176609	0.381017	0	0	0	0	1
rerror_rate	494020	0.057434	0.231624	0	0	0	0	1
srv_rerror_rate	494020	0.057719	0.232147	0	0	0	0	1
same_srv_rate	494020	0.791547	0.38819	0	1	1	1	1
diff_srv_rate	494020	0.020982	0.082206	0	0	0	0	1
srv_diff_host_rate	494020	0.028996	0.142397	0	0	0	0	1
dst_host_count	494020	232.4712	64.7446	0	255	255	255	255
dst_host_srv_count	494020	188.6661	106.0402	0	46	255	255	255
dst_host_same_srv_rate	494020	0.753781	0.41078	0	0.41	1	1	1
dst_host_diff_srv_rate	494020	0.030906	0.109259	0	0	0	0.04	1
dst_host_same_src_port_rate	494020	0.601936	0.481309	0	0	1	1	1
dst_host_srv_diff_host_rate	494020	0.006684	0.042133	0	0	0	0	1
dst_host_serror_rate	494020	0.176754	0.380593	0	0	0	0	1
dst_host_srv_serror_rate	494020	0.176443	0.38092	0	0	0	0	1
dst_host_rerror_rate	494020	0.058118	0.23059	0	0	0	0	1

dst host srv rerror rate	494020	0.057412	0.230141	0	0	0	0	1

3 Data Normalization

As seen from the above continuous variables distribution in the above section the scale of some of the features (columns) values are not in same range as others. This irregularity will make the machine learning model train poorly. Hence the features need to be normalized so that the optimization during model training will happen in a better way and the model will not be sensitive to the features. This research paper uses the existing normalization function that exists in python’s scikit-learn library which can be found here

After applying normalization transformation to each continuous feature, below are the new distribution plots.

4 Modeling

4.1 Model that over-fits the entire dataset

To understand the dataset and to estimate the model size (architecture and hyper-parameters) , we decided to build a model that over-fits the entire dataset. This is an experiment step to understand what model size would be required to overfit entire data. We performed below steps:

➔ Step-1: Using ENTIRE dataset to ”OVERFIT” the model using vannila (single neuron) logistic regression model to reach accuracy 100 percent or close to 100 percent

➔ Step-2: If the accuracy did not reach 100 percent, then a bigger architecture model will be designed and modeled to achieve accuracy of 100 percent or close to 100 percent.

4.1.1 Model-1 Architecture Summary Diagram

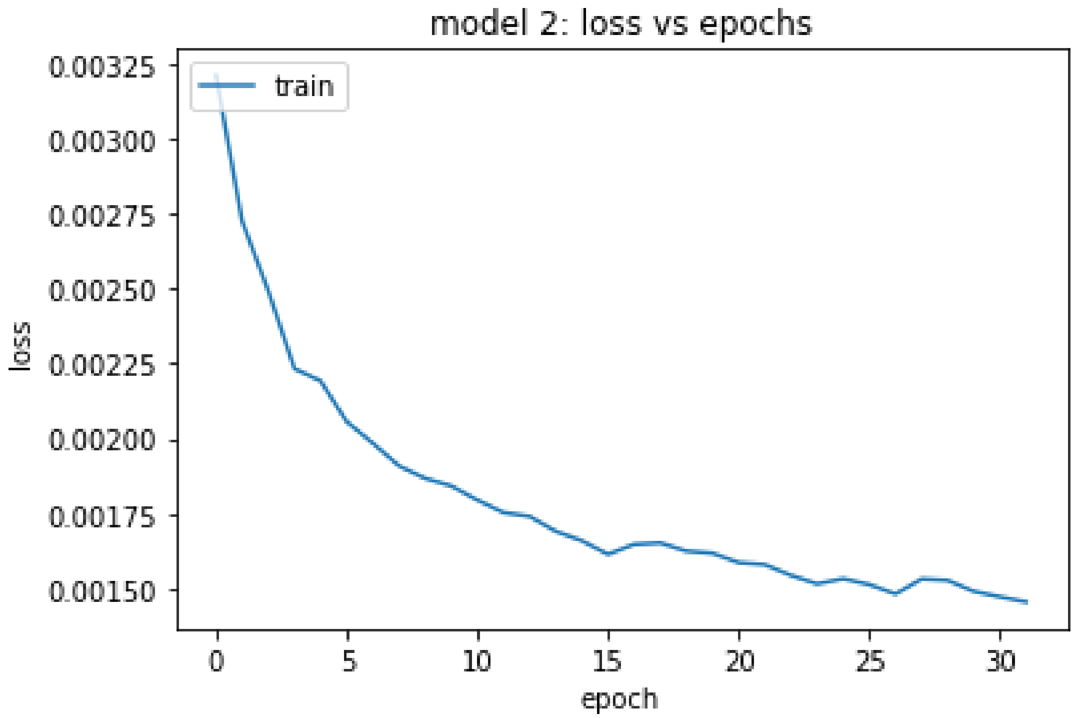
4.1.2 Inference

As seen from the above model, the basic logistic regression model has achieved an accuracy of

99.81 with 256 epochs.A seen from the model-2 performance we have achieved 99.96 percent accuracy (close to 100 percent) which means this model did over-fit the dataset. Therefore, a 4-layer neural network model architecture with 8 neurons, 4 neurons and 2 neurons in internal layers is sufficient enough to overfit the entire dataset.

5 Generalized Model building, selection and evaluation

In the previous section, after exploring the architectures that would result in an over-fitting model, in this section, the focus will be on using a feed-forward neural network (and its architecture variants) to build a best model that would perform well on the validation dataset. The data has been shuffled before performing the experiments and the same dataset has been used for all the models for fair comparison. The code did not implement seeding (which is used for ability to reproduce the model), hence a new run might result in a slight difference in the performances of the models.



5.1 Iteration-0: BaselineModeling

5.2 Iteration-1: Modifying activationfunctions

5.3 Iteration-2: Modifying Optimizerfunctions

5.4 Iteration-3: Modifying BatchSize

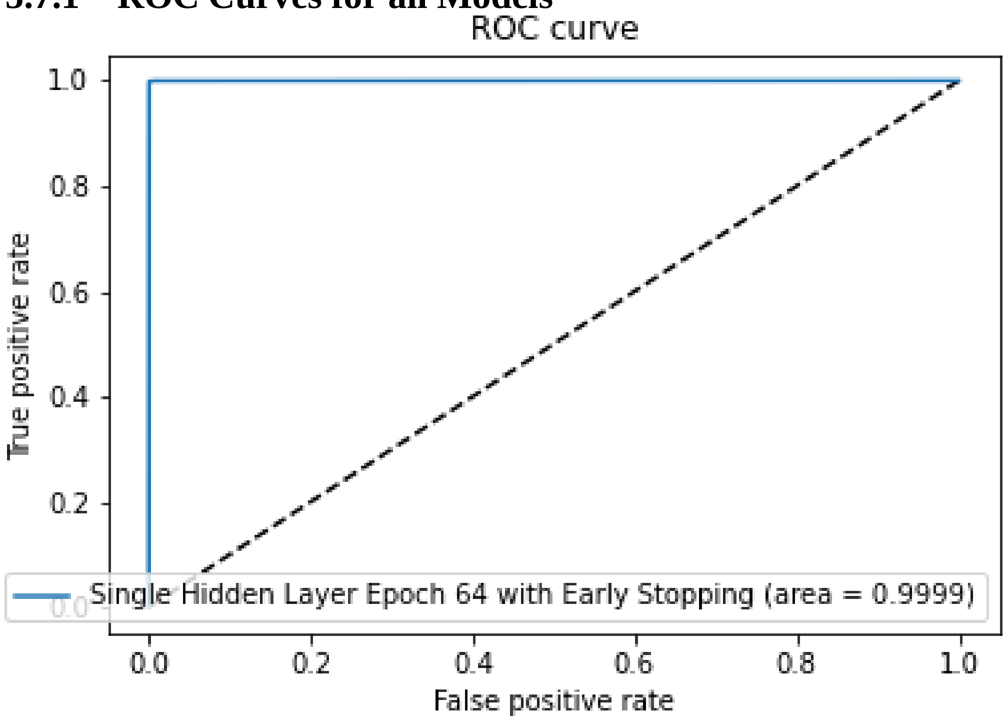
5.5 Iteration-4: Modifying Number ofNeurons

5.6 Iteration-5: Modifying Number ofEpochs

5.7 Model Selection

Model	N_Params	Accuracy	Precision	Recall	F1-Score	auc_score
No Hidden Layers	120	0.9944	0.9932	0.9787	0.9859	0.9984
Single Hidden Layer	485	0.9961	0.994	0.9863	0.9901	0.9992
Two Hidden Layers	493	0.9943	0.9972	0.9746	0.9858	0.9965
Single Hidden Layer ReLU Activation	485	0.9982	0.9942	0.9966	0.9954	0.9999
Single Hidden Layer Softmax Activation	485	0.8034	0	0	0	0.5321
Single Hidden Layer SGD optimizer	485	0.989	0.9886	0.9569	0.9725	0.9955
Single Hidden Layer Adam optimizer	485	0.9948	0.9959	0.978	0.9869	0.9994
Single Hidden Layer Adagrad optimizer	485	0.9842	0.9804	0.9414	0.9605	0.9873
Single Hidden Layer Batch Size 128	485	0.9984	0.9955	0.9963	0.9959	0.9997
Single Hidden Layer Batch Size 64	485	0.9975	0.991	0.9962	0.9936	0.9998
Single Hidden Layer Batch Size 32	485	0.9983	0.9953	0.996	0.9957	0.9998
Single Hidden Layer 2 Neurons	243	0.9931	0.9993	0.9667	0.9828	0.9959
Single Hidden Layer 6 Neurons	727	0.9984	0.9957	0.9959	0.9958	0.9999
Single Hidden Layer Epoch 16	485	0.9988	0.9972	0.9969	0.9971	0.9999
Single Hidden Layer Epoch 32	485	0.999	0.9976	0.9975	0.9975	0.9999
Single Hidden Layer Epoch 64 with Early Stopping	485	0.9991	0.9984	0.9973	0.9978	0.9999
Single Hidden Layer Epoch 64 without Early Stopping	485	0.9991	0.9978	0.9977	0.9978	0.9999

5.7.1 ROC Curves for all Models



6 Overfitting with target variable

7 Custom predict function

8 Iterative Feature Reduction

8.1 Feature Importance

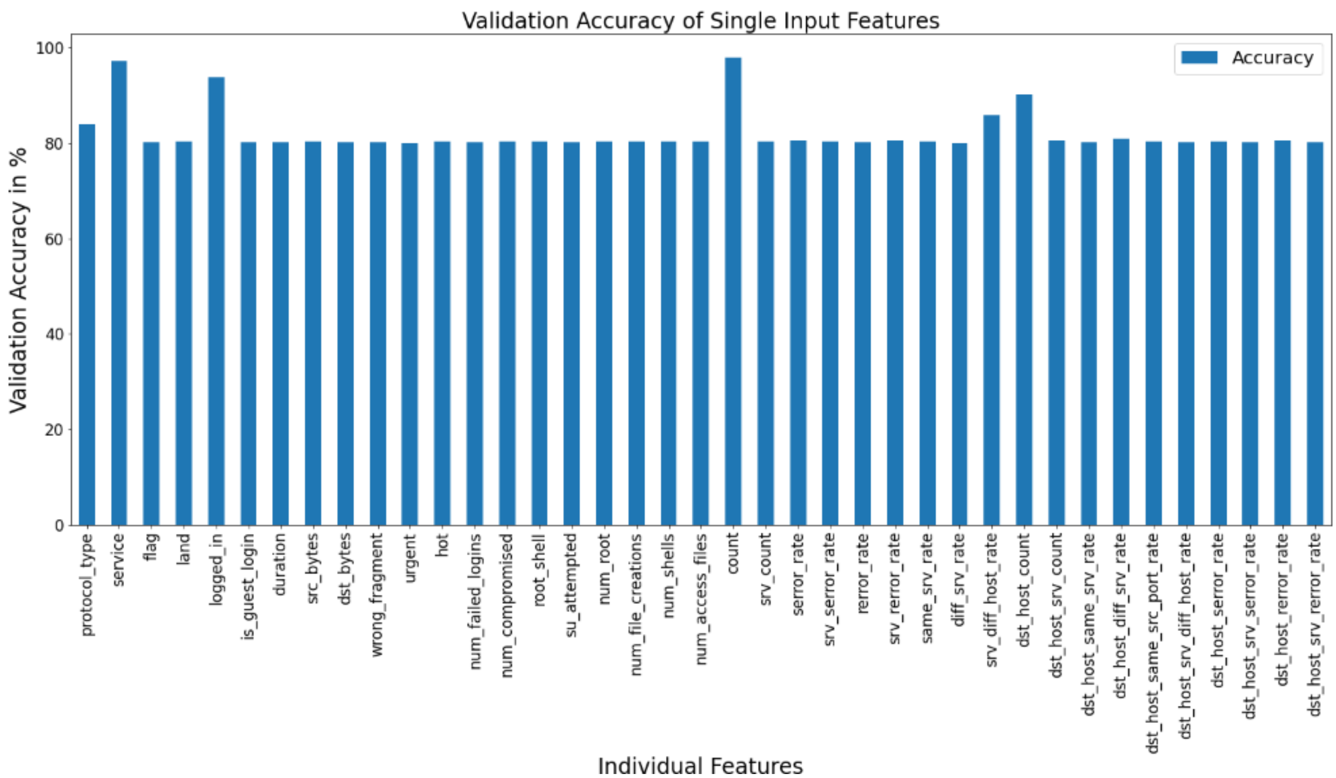
We ran all 39 features through the best model architecture individually and obtained the accuracy of each model on the validation set. We then ranked the model performances based on the validation accuracy. Our assumption is that feature with lowest validation accuracy has less feature importance compared to others.

As seen in the below table and the picture, the feature ”count” has the highest importance and the feature ”diff-srv-rate” has the least importance.

Table 3: Feature Importance Ranking

Rank	Accuracy	feature_name
0	79.959315	diff_srv_rate
1	80.054450	urgent
2	80.109107	dst_host_srv_diff_host_rate
3	80.175906	dst_host_same_srv_rate
4	80.178940	su_attempted
5	80.182987	duration
6	80.184001	is_guest_login
7	80.204242	flag
8	80.215377	dst_host_srv_serror_rate
9	80.223471	num_failed_logins
10	80.252826	wrong_fragment
11	80.259907	dst_bytes
12	80.271041	rerror_rate
13	80.271041	dst_host_srv_rerror_rate
14	80.283189	dst_host_serror_rate
15	80.285209	same_srv_rate
16	80.288249	land
17	80.299383	srv_count
18	80.302417	srv_serror_rate
19	80.305451	num_root
20	80.330753	root_shell
21	80.340874	num_shells
22	80.341887	num_file_creations
23	80.378324	num_access_files
24	80.385411	src_bytes
25	80.407673	num_compromised
26	80.437028	dst_host_same_src_port_rate
27	80.444109	hot
28	80.465364	srv_rerror_rate
29	80.479532	dst_host_srv_count
30	80.494720	serror_rate
31	80.529130	dst_host_rerror_rate
32	80.915755	dst_host_diff_srv_rate
33	83.928788	protocol_type
34	85.939842	srv_diff_host_rate
35	90.297961	dst_host_count
36	93.734062	logged_in

37	97.144854	service
38	97.977817	counts



8.2 Performance After Removal

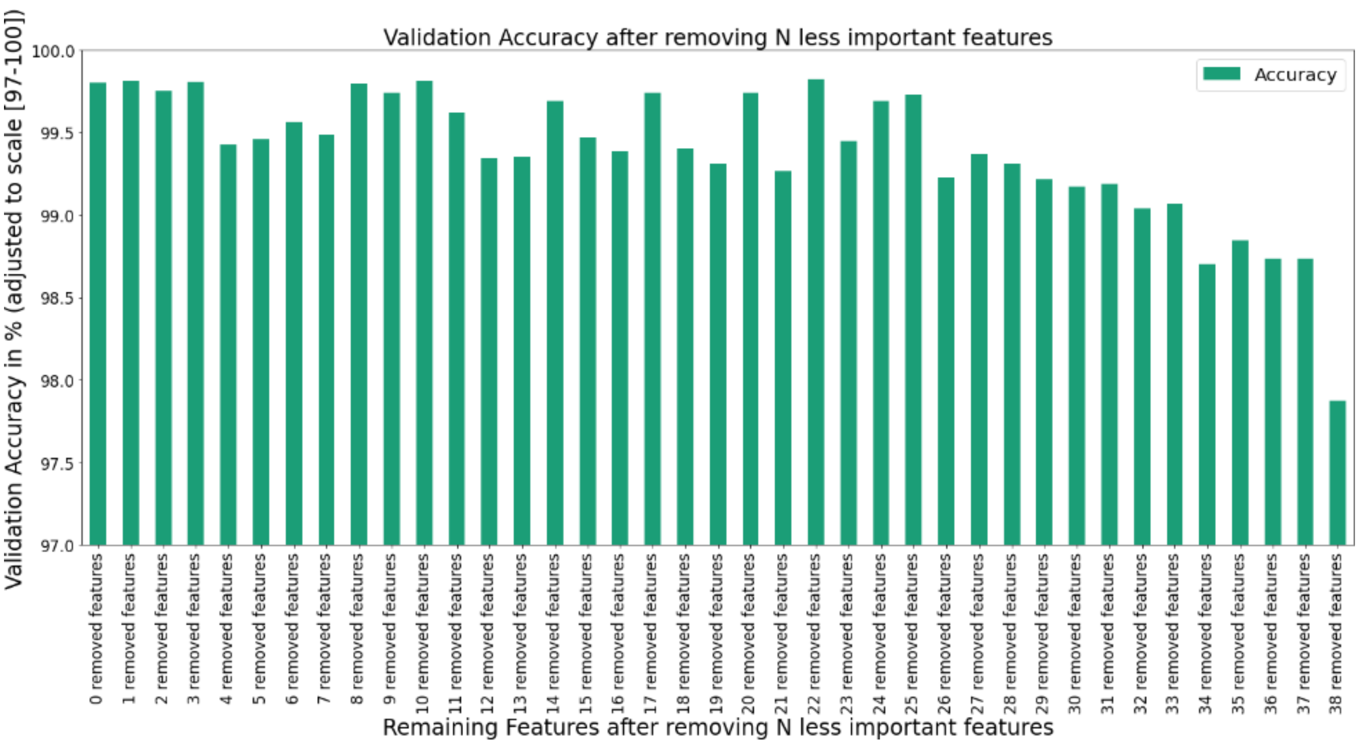
The list of features used for this data set can be categorized into three categories - 1) Basic features of individual TCP connections.2) Content features within a connection suggested by domain knowledge. 3) Traffic features computed using a two-second time window.

We observed if there is a correlation between above three categories and the feature importance values of the individual features. We did not find any strong correlation between this domain specific categorization. However, it is evident that the mostly traffic related features ended up having high feature importance values. Therefore, we strongly believe that the actual values in each feature contributed to the model performance which made sense.

Based on the above feature importance values, we started building the models by removing the unimportant features (importance based on the validation accuracy, the higher the better) an iterative way. As seen in the below plot, all models' validation accuracy values are obtained and are plotted below.

The main motive of this experiment is to find out the ideal subset of features that would yield best results with less computation power consumption and modeling time. From the below plot, we concluded that the model built after removing 22 less important features showed higher validation accuracy. We thus select the features used for this model as the best features for this data set. The features used for this model are:

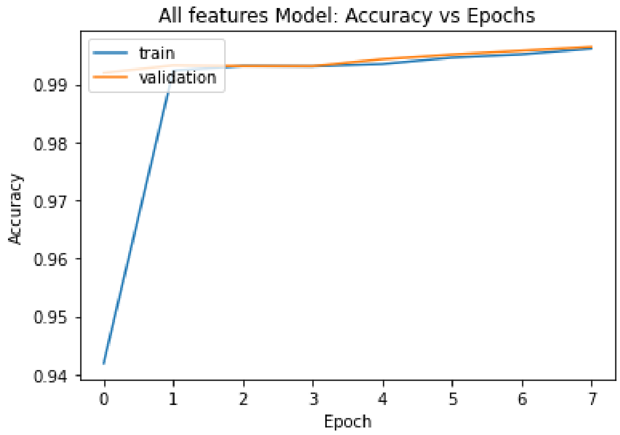
['num access files', 'src bytes', 'num compromised', 'dst host_same_src port rate', 'hot', 'srv rerror rate', 'dst host srv_count', 'serror rate', 'dst host error rate', 'dst-host-diff-srv-rate', 'protocol-type', 'srv-diff-host-rate', 'dst-host-count', 'logged-in', 'service', 'count']

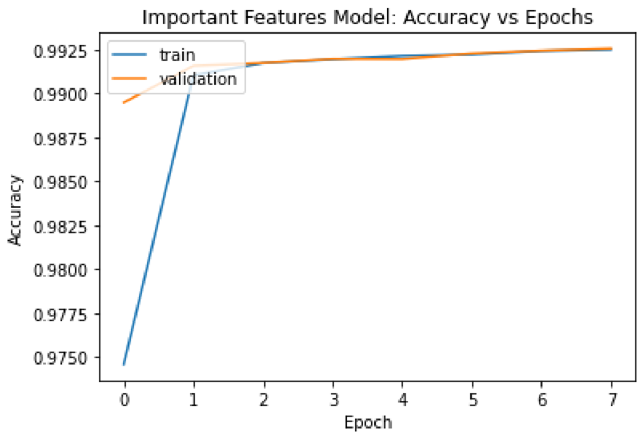


8.3 Performance Comparison: All-Features-Model vs Important-Features-Model

We performed comparison analysis on the model that was built with all features and on the model that was built with important features. As seen in the below plots, the learning rate of the model that was built with all features is not increasing and saturated right after first epoch. Whereas, the learning rate of the model built with subset of importance features (taken from previous step), increased per each epoch (loss reduced for every epoch). Therefore, it is evident that the model built with subset of important features showed more generalization which willbe helpful when the data set sizes are varied.

Moreover, building the model with less number of features requires less computations thus less computing power. The model built with the above subset of important features is evidently better in terms of modeling times, model performance and computation cost.





9 Conclusion

After using iterative feature reduction technique, features that are derived as the important ones for this data set are 'dst_bytes', 'same_srv_rate', 'logged_in', 'is_guest_login', 'dst_host_diff_srv_rate', ‘root_shell', 'num_file_creations', 'src_bytes', 'duration', 'dst_host_srv_count', 'dst_host_srv_diff_host_rate', 'protocol_type', 'srv_diff_host_rate', 'dst_host_count', 'service','count'

References

[1] Manoj Kumar Putchala. Deep learning approach for intrusion detection system (ids) inthe internet of things (iot) network using gated recurrent neural networks (gru). 2017.

[2] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. IEEE, 2009.

[3] Steven Huang. Kdd cup 1999 data, computer network intrusion detection (version 1). 2018. Available from https://www.kaggle.com/galaxyh/kdd-cup-1999-data.