# ML/DL Driven Intelligent Intrusion Detection Approach

CSE 534 Final Project Report

Akshay Somayaji S
SBU ID# 113322316

Adith Kumar Bussi
SBU ID# 113217801

GitHub Repo: https://github.com/chillaks/CSE-534-Project

## Introduction and Background

Intrusion detection is a crucial task in information security. With the ever-growing need to secure data in IT infrastructures and new exploits being crafted by attackers every day, it is imperative that the defense mechanisms too are up to-date with their counter measures. **Network-based Intrusion Detection Systems (NIDS)** are devices intelligently distributed within networks that passively inspect traffic traversing the devices on which they sit to detect malicious traffic on a network.

Such systems presently exist in various variants but can be largely categorized into two broad classifications; *signature-based* and *anomaly-based* detection, depending on their approaches to recognize attack packets. The signature-based approach uses well-known fingerprints of the attack packets to detect breaches in secured networks. Thus, new attacks with unknown signatures can potentially get by undetected. Anomaly-based systems, on the other hand, uses ML to create a defined model of trustworthy activity, and then compare new behavior against this trust model. Anomaly-based NIDS is hence a major research area, with novel Machine and Deep Learning algorithms being introduced on a regular basis.

## Problem Statement

The aim of this project is to evaluate different ML and deep learning techniques to detect any abnormal traffic within the network, including ones that have not been encountered before. Our problem statement hence boils down to a classification task, where we will first implement binary classifiers to determine normal/attack packets, and then a multi-class classifier to differentiate normal and specific types of attacks. In this report, we will be describing three of the primary tasks that we have completed, which we will see in detail in the results section. They are as follows:

1. Evaluate the methodology and results documented in the paper [1], and verify if autoencoders outperform other ML models (Main task)
2. Explore other recent classifiers outside of the paper that may perform better. (Additional task)
3. Identify the key features of our dataset that have the highest impact on every class of attack type, the presence of which can thus be used as a possible indicator of that attack. (Additional task)

# The Dataset

We build and evaluate our models using the NSL-KDD dataset[2], which is widely used as the benchmark for testing several intrusion detection systems. The dataset contains 140000 labeled network data with 4 different attack classes: Denial of Service (**DoS**), Root to Local (**R2L**), User to Root (**U2R**) and Probing (Fig 1), which are explained further below:

- DoS is an attack that tries to shut down traffic flow to and from the target system. The IDS is flooded with an abnormal amount of traffic, which the system can't handle, and shuts down to protect itself.
- Probe or surveillance is an attack that tries to get information from a network. The goal here is to act like a thief and steal important information, whether it be personal information about clients or banking information.
- U2R is an attack that starts off with a normal user account and tries to gain access to the system or network, as a super-user (root)
- R2L is an attack that tries to gain local access to a remote machine. An attacker does not have local access to the system/network and tries to "hack" their way into the network.

| Dataset | Number of Records: | | | | | |
|---|---|---|---|---|---|---|
| | Total | Normal | DoS | Probe | U2R | R2L |
| KDDTrain+20% | 25192 | 13449 (53%) | 9234 (37%) | 2289 (9.16%) | 11 (0.04%) | 209 (0.8%) |
| KDDTrain+ | 125973 | 67343 (53%) | 45927 (37%) | 11656 (9.11%) | 52 (0.04%) | 995 (0.85%) |
| KDDTest+ | 22544 | 9711 (43%) | 7458 (33%) | 2421 (11%) | 200 (0.9%) | 2654 (12.1%) |

**Fig 1: Split of attack classes in the dataset**

There are 41 features in the dataset (Fig 2), which can broadly be categorized into the below categories:

- Features 1- 9: Intrinsic features, can be derived from the header of the packet.
- Features 10 – 22: Content features, hold information about the original packets.
- Features 23 – 31: Time-based features, hold the analysis of the traffic input over a two-second window
- Features 32 – 41: Host-based features, hold the analysis of the traffic input over a series of connections made

The feature types in this data set can be broken down into 4 types:

- 4 Categorical (Features: 2, 3, 4, 42)
- 6 Binary (Features: 7, 12, 14, 20, 21, 22)
- 23 Discrete (Features: 8, 9, 15, 23–41, 43)
- 10 Continuous (Features: 1, 5, 6, 10, 11, 13, 16, 17, 18, 19)

The last 2 columns are the labels (sub-classes of attack types) and the difficulty score. The label is what we use as the target variable for our models.

| No. | Feature Name | No. | Feature Name |
|---|---|---|---|
| 1 | Duration | 22 | is_guest_login |
| 2 | protocol type | 23 | **count** |
| 3 | service | 24 | srv_count |
| 4 | **flag** | 25 | **serror_rate** |
| 5 | src_bytes | 26 | **srv_serror_rate** |
| 6 | **dst_bytes** | 27 | **rerror_rate** |
| 7 | land | 28 | srv_rerror_rate |
| 8 | wrong_fragment | 29 | **same_srv_rate** |
| 9 | urgent | 30 | diff_srv_rate |
| 10 | hot | 31 | srv_diff_host_rate |
| 11 | num_failed_logins | 32 | dst_host_count |
| 12 | logged_in | 33 | **dst_host_srv_count** |
| 13 | num_compromised | 34 | dst_host_same_srv_rate |
| 14 | root_shell | 35 | dst_host_diff_srv_rate |
| 15 | su_attempted | 36 | dst_host_same_src_port_rate |
| 16 | num_root | 37 | dst_host_srv_diff_host_rate |
| 17 | num_file_creations | 38 | **dst_host_serror_rate** |
| 18 | num_shells | 39 | **dst_host_srv_serror_rate** |
| 19 | num_access_files | 40 | dst_host_rerror_rate |
| 20 | num_outbound_cmds | 41 | dst_host_srv_rerror_rate |
| 21 | is_host_login | 42 | |

**Fig 2: Feature variables of the NSL-KDD dataset**

# Approach

We follow the standard set of steps (Fig 4) as with any other data-science related task. Below is the sequence followed before building the model:

1. **Data Pre-Processing:**
    - Outlier detection and removal – Use Median Absolute Deviation Estimator (MADE) to remove columns containing more than 80% of rows as 0 values.
    - Due to limited data for **U2R** attack type, we drop all such rows from the dataset.
    - Data Normalization:
        - Use Min-Max Normalization for continuous features
        - Use One Hot Encoding for categorical features (Fig 3)

| Protocol Type (2) | Service (3) | | | | Flag (4) |
|---|---|---|---|---|---|
| • icmp<br>• tcp<br>• udp | • other<br>• link<br>• netbios_ssn<br>• smtp<br>• netstat<br>• ctf<br>• ntp_u<br>• harvest<br>• efs<br>• klogin<br>• systat<br>• exec<br>• nntp<br>• pop_3<br>• printer<br>• vmnet<br>• netbios_ns | • urh_i<br>• ssh<br>• http_8001<br>• iso_tsap<br>• aol<br>• sql_net<br>• shell<br>• supdup<br>• auth<br>• whois<br>• discard<br>• sunrpc<br>• urp_i<br>• Rje<br>• ftp<br>• daytime<br>• domain_u<br>• pm_dump | • time<br>• hostnames<br>• name<br>• ecr_i<br>• bgp<br>• telnet<br>• domain<br>• ftp_data<br>• nnsp<br>• courier<br>• finger<br>• uucp_path<br>• X11<br>• imap4<br>• mtp<br>• login<br>• tftp_u<br>• kshell | • private<br>• http_2784<br>• echo<br>• http<br>• ldap<br>• tim_i<br>• netbios_dgm<br>• uucp<br>• eco_i<br>• Remote_job<br>• IRC<br>• http_443<br>• red_i<br>• Z39_50<br>• Pop_2<br>• gopher<br>• Csnet_ns | • OTH<br>• S1<br>• S2<br>• RSTO<br>• RSTRs<br>• RSTOS0<br>• SF<br>• SH<br>• REJ<br>• S0<br>• S3 |

**Fig 3: Categorical Features that are converted to feature vectors after One Hot Encoding**


**Fig 4: IDS Model Evaluation Pipeline**

## 2. Data Labeling:

In Fig 4 and Fig 5, we can see the proportion of attack and normal types that will later be used to build both the Binary and Multi-class Classifiers.

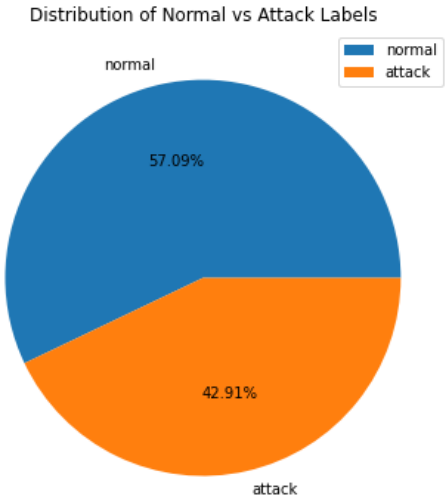



**Fig 5: Binary Classification**

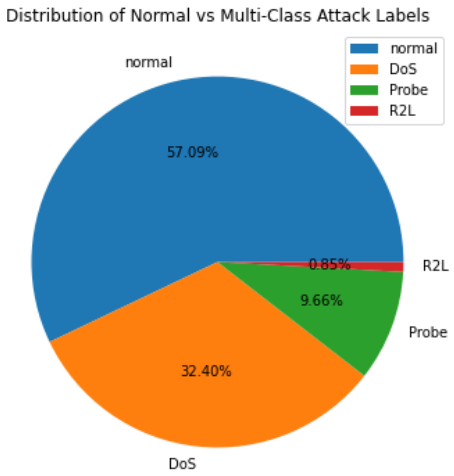



**Fig 6: Multi-class Classification**

## 3. Feature Selection:

We filter out all the features that have low correlation with the label by using the SelectKBest selector. We can find the highest weight features below, which we will be using to build our models:

```
pd.Series(fs.scores_, index=discrete).sort_values(ascending = False)
```

```
same_srv_rate                40232.920225
dst_host_srv_serror_rate     16715.980137
dst_host_serror_rate         16088.460097
srv_serror_rate              15956.452007
serror_rate                  15785.989942
dst_host_same_srv_rate       11436.901521
dst_host_srv_diff_host_rate   9847.313194
dst_host_srv_count            7911.890531
count                         7379.363178
srv_count                     5979.052315
dst_host_same_src_port_rate   5401.084902
dst_host_diff_srv_rate        4499.427790
dst_host_count                3264.372113
diff_srv_rate                 3197.355436
srv_diff_host_rate            2137.170039
dtype: float64
```

**Fig 6: Highest correlating features features**

4. **Model Building:**

We build the model using the Autoencoder technique and evaluate it against other popular machine / deep learning methods to see if it actually performs better as stated in the paper. Autoencoders are an unsupervised Deep Learning technique that learns the pattern of a normal process. Anything that does not follow this pattern is classified as an anomaly. They consist of two modules:

I. *Encoder*: Transforms the input data vector into a lower dimension
II. *Decoder*: Recreates the original data from the underlying features that are embedded in the compressed vectors.

Additionally, we have also used Random Forest and Decision Trees (Fig 7) classifiers to evaluate their performance with the dataset. A detailed description of this comparison and evaluation metrics can be found in the Results section.

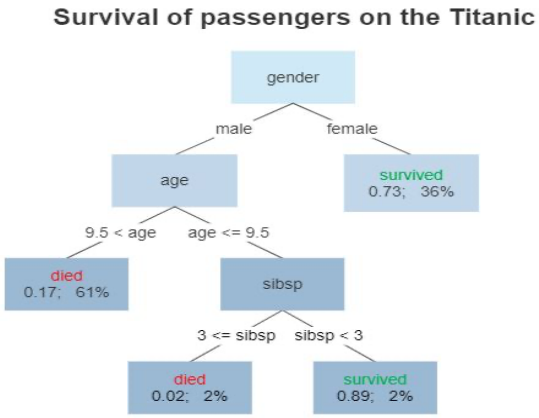



**Fig 7: Decision Trees**

# Results

**1. Building and comparing models presented in the paper.**

For the first part of our project, we build the models and run the experiments as mentioned in the paper and compare our results with the one presented by the authors. The models are trained with the preprocessed data to classify network packets into 2 classes – Attack and Normal.

We follow the similar preprocessing steps as mentioned in the paper and present the obtained accuracies.

- Binary Classification

| Model | Presented Accuracy | Obtained Accuracy |
|---|---|---|
| **AE** | 84.21% | 79.78% |
| **LSTM** | 82.04% | 78.21% |
| **MLP** | 81.65% | 79.13% |
| **LSVM** | 80.8% | 78.74% |
| **QSVM** | 83.15% | 79.4% |

- Multi Class Classification

| Model | Presented Accuracy | Obtained Accuracy |
|---|---|---|
| **AE** | 87% | 72.24% |
| **LSTM** | 80.67% | 73.56% |
| **MLP** | 81.43% | 80.77% |
| **LSVM** | 81.4% | 77.34% |
| **QSVM** | 83.65% | 60.69% |

## 2. Exploring the Blackbox

The paper presents the results in terms of accuracy and other statistical numbers, but it fails to provide insights into the network features that are most important to classify the network data.
Main objective of our project was to identify and learn the important network features, so our efforts were concentrated on diving into the Blackbox and gaining insights on network features which help the Machine Learning models internally to classify the packets into different classes.

We built 2 new classifiers, DecisionTree and RandomForest. As explained in the modelling section, Decision Tree classifies using a hierarchy of features in form of a tree. The features which are closest to the root are of the highest importance and the most decisive. RandomForest is an ensemble of DecisionTrees which constructs multiple DecisionTrees and pools together the results to classify the data. This makes them the ideal models to train to identify network features. DecisionTree had an accuracy of 78% for multiclass classification and RandomForest was able to achieve 81%.

DecisionTree and RandomForest can provide the weights learned on each feature using the *feature_importances_* method. The top 5 features are as shown below.

- DecisionTree

| | |
|---|---|
| **flag_SF** | 0.5042174285 |
| **dst_host_serror_rate** | 0.1631269608 |
| **service_domain_u** | 0.08479269121 |
| **service_ftp** | 0.06442231693 |
| **srv_count** | 0.04391217008 |

RandomForest :

| | |
|---|---|
| **flag_SF** | 0.1388173254 |
| **flag_SH** | 0.1013037257 |
| **flag_RSTOS0** | 0.05530383838 |
| **diff_srv_rate** | 0.05362965763 |
| **dst_host_serror_rate** | 0.05043431042 |

Although this gives us insights into the top network features that are used by the model to classify the entirety of the data, it fails to build an intuition which about distinctive features that help to identify a particular class of attack. For example, it would be ideal to find out which network aspect is a unique feature of DDoS attack and helps to pick out DDoS packets from the group. That is why, we use additional techniques to pinpoint distinctive features that correlates to a particular class of data.

## 3. Diving Deeper into the Blackbox

As we need to target individual features, we plan to use OneVsRest classifiers which are trained to separate one class from the rest of the group. In our OneVsRest classifier group, we use 4 DecisionTrees, one for each class which learns to pick out that particular class from the rest.

The top 5 features of each are as follows:

| | Dos |
|---|---|
| dst_host_serror_rate | 0.625830 |
| service_domain_u | 0.103884 |
| diff_srv_rate | 0.078954 |
| same_srv_rate | 0.073636 |
| flag_SF | 0.045652 |

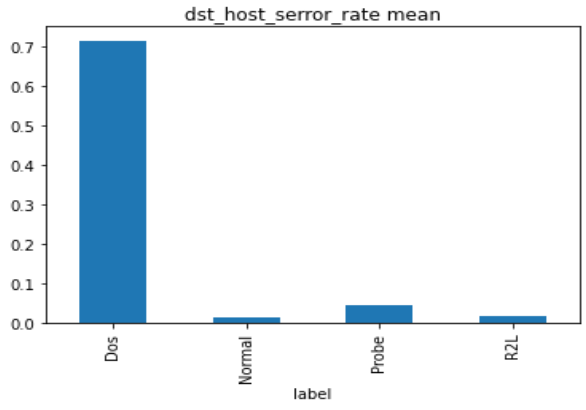
| | Probe |
|---|---|
| dst_host_diff_srv_rate | 0.309893 |
| service_courier | 0.301389 |
| flag_SF | 0.284453 |
| service_http | 0.038314 |
| dst_host_same_srv_rate | 0.013838 |

| | R2L |
|---|---|
| dst_host_same_src_port_rate | 2.960790e-01 |
| flag_SH | 1.379910e-01 |
| flag_SF | 1.349142e-01 |
| dst_host_srv_count | 1.331199e-01 |
| dst_host_srv_diff_host_rate | 1.144253e-01 |

| | Normal |
|---|---|
| flag_SF | 0.712434 |
| service_domain_u | 0.093938 |
| service_ftp | 0.064885 |
| dst_host_same_srv_rate | 0.049646 |
| flag_SH | 0.030285 |

- We verify if these features are unique in some manner to its respective column by using Data Analysis techniques on our Dataset.
- We observe that there is an excellent correlation between the features output by the model and the data.
- We plot graphs with the top feature of each class to visually explain the distinctiveness of the feature.

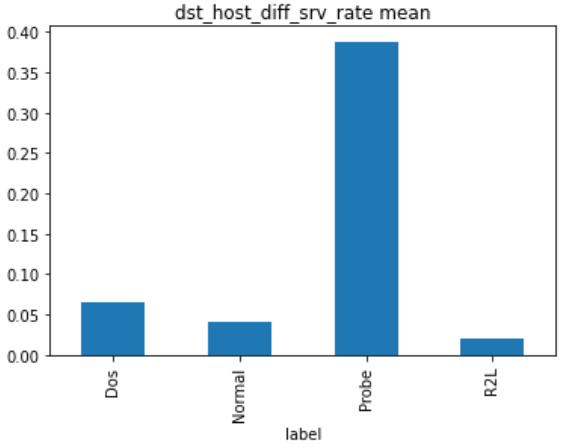
### 1. DoS




- *dst_host_serror_rate* : The percentage of connections that have activated the flag s0, s1, s2 or s3, among the connections having the same destination host IP address. These flags are activated when the connections weren't established and terminated in a smooth way without any error. Further explanation of each flag is available in [3].

- We can observe that the mean value of *dst_host_serror_rate* is considerably high for Dos packets compared to other classes.

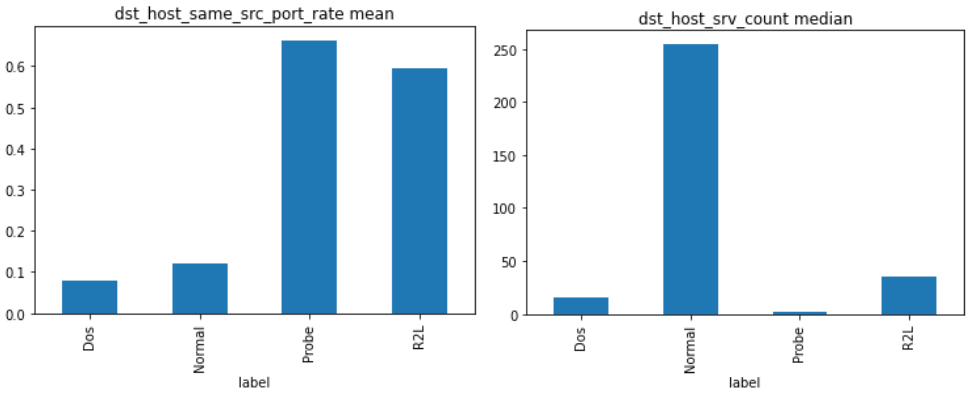
## 2. Probe




- *dst_host_diff_srv_rate* : The percentage of connections that were to different services, among the connections having the same destination host IP address.
- We can observe that the mean value of *dst_host_diff_srv_rate* is considerably high for Probe packets compared to other classes.

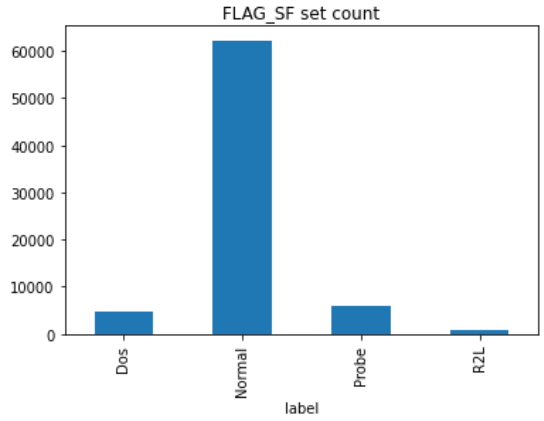
## 3. R2L

- In the case of R2L, the top feature was not alone enough to clearly identify the R2L class.
- The first feature *dst_host_same_src_port_rate* separates both Probe and R2L packets as it has a high mean value.
- We consider another top feature *dst_host_srv_count,* which further divides Probe and R2L values has they have very different medians.



4. **Normal**

- Normal class could be clearly spotted with many features. The top feature flag_SF is set when the connection establishment and termination happens without any error.




# Conclusion

We were able to successfully detect intrusion and classify network packets into their respective type of attack. The accuracies of different ML models are presented and MLP and RandomForest models gave the best accuracy among the lot. Further, we delved deep into the trained ML models to extract useful information on importance of each feature which correlates with a particular classification. OneVsRest classifier consisting of 4 DecisionTree models was used, and top features of each class was obtained. We verified the importance of top features by visualizing the mean of that feature using graphs on our Dataset.

# References

1. A Novel Statistical Analysis and Autoencoder Driven Intelligent Intrusion Detection Approach
https://sci-hubtw.hkvisa.net/10.1016/j.neucom.2019.11.016
https://doi.org/10.1016/j.neucom.2019.11.016

2. The NSL-KDD dataset https://www.unb.ca/cic/datasets/nsl.html

3. A Deeper Dive into the NSL-KDD Data Set https://towardsdatascience.com/a-deeper-dive-into-the-nsl-kdd-data-set-15c753364657

4. https://en.wikipedia.org/wiki/Decision_tree_learning