INCREMENT - II

CSCE 5210: Fundamentals of Artificial Intelligence

Intrusion Detection System

Github: https://github.com/fouzanuddin/IntrusionDetection

Members:
1. Fouzan uddin - 11475342
2. Syed Araib Karim - 11479107
3. Shabab Alghamdi - 11099275
4. Mohammed Abdul Moin - 11508271

Abstract:
A network-based intrusion detection system (NIDS) monitors a network for harmful traffic. In order to evaluate all traffic, including all unicast traffic, NIDS typically require promiscuous network access. NIDS are non-interfering devices that monitor traffic without interfering with it.

A dataset including a wide range of intrusions simulated in a military network environment was supplied for auditing. By mimicking a typical US Air Force LAN, it established an environment in which raw TCP/IP dump data for a network could be acquired. The LAN was focused as if it were a real setting, and various attacks were launched. A connection is a series of TCP packets that begin and stop at a specific time interval and allow data to flow from a source IP address to a target IP address using a well-defined protocol. In addition, each link is classified as either normal or an attack, with only one attack kind. Each connection record is around 100 bytes long.

Attacks fall into four main categories:
- DOS: denial-of-service, e.g. syn flood;
- R2L: unauthorized access from a remote machine, e.g. guessing password;
- U2R: unauthorized access to local superuser (root) privileges, e.g., various ``buffer overflow'' attacks; probing: surveillance and others
- probing, e.g., port scanning.

1. Introduction:

1. Problem specification:
   The Intrusion Detection System (IDS) is a detective tool that detects harmful (including policy-violating) behavior. An Intrusion Prevention System (IPS) is basically a preventive device that can both detect and block hostile activity. We can simulate an attack through the dataset and create a machine learning model to find abnormal events which can be classified as an attack.

2. Dataset:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	l
0	0	tcp	ftp_data	SF	491	0	0	0	0	0	0	
1	0	udp	other	SF	146	0	0	0	0	0	0	
2	0	tcp	private	S0	0	0	0	0	0	0	0	
3	0	tcp	http	SF	232	8153	0	0	0	0	0	
4	0	tcp	http	SF	199	420	0	0	0	0	0	

The dataset contains the following features:

#	Column	Non-Null Count	Dtype
0	duration	20153 non-null	int64
1	protocol_type	20153 non-null	object
2	service	20153 non-null	object
3	flag	20153 non-null	object
4	src_bytes	20153 non-null	int64
5	dst_bytes	20153 non-null	int64
6	land	20153 non-null	int64
7	wrong_fragment	20153 non-null	int64
8	urgent	20153 non-null	int64
9	hot	20153 non-null	int64
10	num_failed_logins	20153 non-null	int64
11	logged_in	20153 non-null	int64
12	num_compromised	20153 non-null	int64
13	root_shell	20153 non-null	int64
14	su_attempted	20153 non-null	int64
15	num_root	20153 non-null	int64
16	num_file_creations	20153 non-null	int64
17	num_shells	20153 non-null	int64
18	num_access_files	20153 non-null	int64
19	is_host_login	20153 non-null	int64
20	is_guest_login	20153 non-null	int64
21	count	20153 non-null	int64
22	srv_count	20153 non-null	int64
23	serror_rate	20153 non-null	float64
24	srv_serror_rate	20153 non-null	float64
25	rerror_rate	20153 non-null	float64
26	srv_rerror_rate	20153 non-null	float64
27	same_srv_rate	20153 non-null	float64
28	diff_srv_rate	20153 non-null	float64
29	srv_diff_host_rate	20153 non-null	float64
30	dst_host_count	20153 non-null	int64
31	dst_host_srv_count	20153 non-null	int64
32	dst_host_same_srv_rate	20153 non-null	float64
33	dst_host_diff_srv_rate	20153 non-null	float64

```
34  dst_host_same_src_port_rate  20153 non-null  float64
35  dst_host_srv_diff_host_rate  20153 non-null  float64
36  dst_host_serror_rate         20153 non-null  float64
37  dst_host_srv_serror_rate     20153 non-null  float64
38  dst_host_rerror_rate         20153 non-null  float64
39  dst_host_srv_rerror_rate     20153 non-null  float64
```

feature name	*description*	*type*
duration	length (number of seconds) of the connection	continuous
protocol_type	type of the protocol, e.g. tcp, udp, etc.	discrete
service	network service on the destination, e.g., http, telnet, etc.	discrete
src_bytes	number of data bytes from source to destination	continuous
dst_bytes	number of data bytes from destination to source	continuous
flag	normal or error status of the connection	discrete
land	1 if connection is from/to the same host/port; 0 otherwise	discrete
wrong_fragment	number of ``wrong'' fragments	continuous
urgent	number of urgent packets	continuous

figure 1

figure 2

feature name	*description*	*type*
hot	number of ``hot'' indicators	continuous
num_failed_logins	number of failed login attempts	continuous
logged_in	1 if successfully logged in; 0 otherwise	discrete
num_compromised	number of ``compromised'' conditions	continuous
root_shell	1 if root shell is obtained; 0 otherwise	discrete
su_attempted	1 if ``su root'' command attempted; 0 otherwise	discrete
num_root	number of ``root'' accesses	continuous
num_file_creations	number of file creation operations	continuous
num_shells	number of shell prompts	continuous
num_access_files	number of operations on access control files	continuous
num_outbound_cmds	number of outbound commands in an ftp session	continuous
is_hot_login	1 if the login belongs to the ``hot'' list; 0 otherwise	discrete
is_guest_login	1 if the login is a ``guest''login; 0 otherwise	discrete

3. Problem analysis:
   We sought to create a simple machine learning approach for solving intrusion detection, which can be treated as a binary classification issue, by employing classifiers. That is, the goal is to determine whether the network traffic is abnormal behavior or not.
   The goal of an IDS is to identify different kinds of malicious network traffic and computer usage, which cannot be identified by a traditional firewall. Failure to prevent intrusions could jeopardize security services' credibility, such as data confidentiality, integrity, and availability.

2. Design and Milestones:

1. Data Preprocessing: It is a crucial phase in the data mining process that involves manipulating or removing data before it is utilized to ensure or improve performance.
2. Dataset Split: Dataset splitting becomes necessary in ML algorithms to remove bias in training data. When parameters of a machine learning algorithm are changed to best suit the training data, the consequence is frequently an overfit algorithm that performs badly on actual test data.
3. Exploratory Data Analysis (EDA): Exploratory Data Analysis, or EDA, is a technique for extracting information from data. Using statistical graphs and other visualization tools, Data Scientists and Analysts aim to uncover diverse patterns, relationships, and anomalies in the data. The following items are included in EDA: Get the most out of a data set.
4. KFold Validations: Cross-validation is a resampling technique for evaluating machine learning models on a small sample of data. The process includes only one parameter, k, which specifies the number of groups into which a given data sample should be divided.Here we use a baseline Random Forest Model.
5. Making Predictions:The technique of applying data analytics to create predictions based on data is known as predictive analytics. This procedure creates a predictive model for forecasting future events by combining data with analysis, statistics, and machine learning techniques.

2.1 Proposed Method:

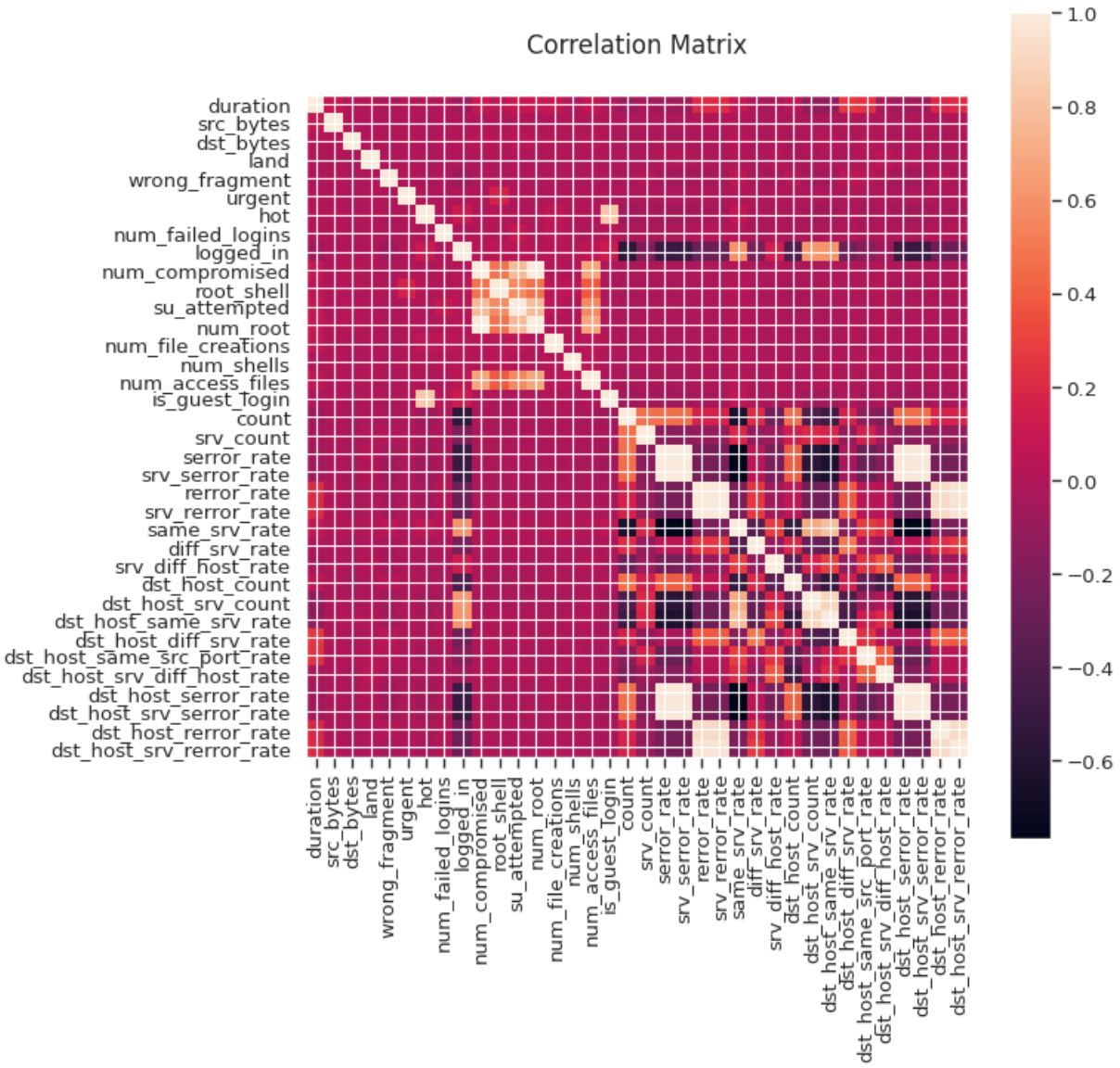
- Data Preprocessing
- EDA
- Feature Importance
- Making a simple Baseline Random Forest model
- A Comparative analysis of multiple models on the dataset.
- Model Selection
- Test on unseen Dataset

2.2 Data processing:

1. First, we generate Basic information about the training data shown, including:
    - Data appearance
    - Data types of features
    - NaN ratio of each features
    - Stats of features

2. We try to make sense of the categorical features. There are three categorical features in the dataset, including protocol type, service and flag.

3. figure 3



3. Finding correlation of the train dataset.

4. We try to find ratios of each unique value.

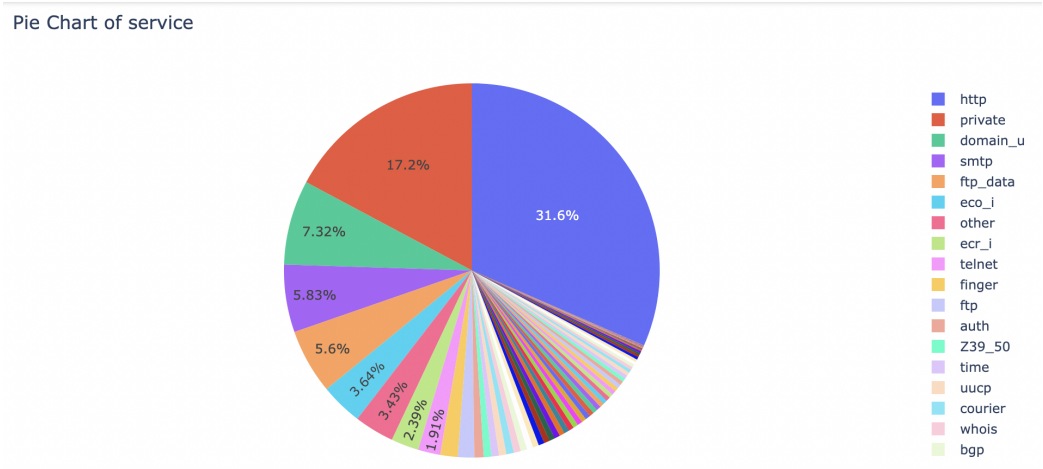
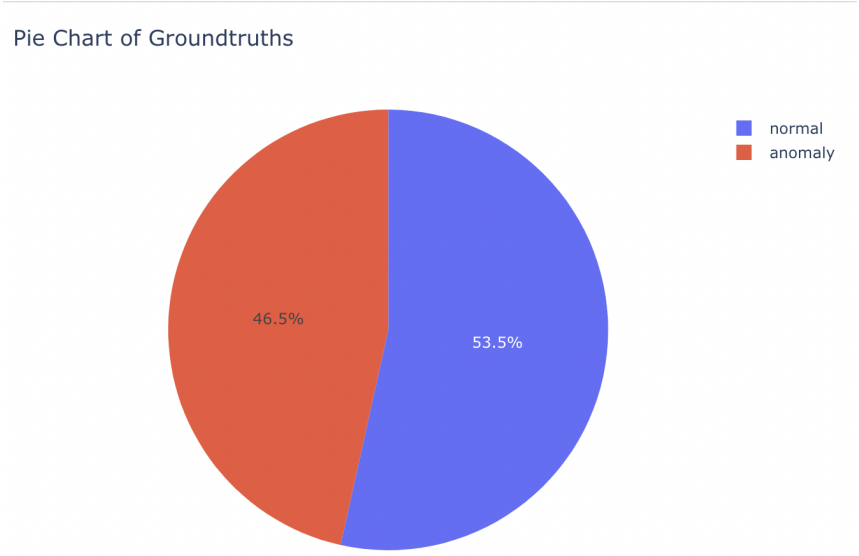


figure 4

5. We try to find distribution of these categorical features in the dataset
6. We plot univariate histograms of the features to give us preliminary understanding about the distributions.
7. We generate the proportion of the value with the most count in each feature - from the perspective of value count.
8. The variance of each feature - from the perspective of value.
9. We generate a distribution of the ground truths to see if there's a problem of Class Imbalance.



10. figure 5

2.3 Experimental Settings:

- We are already given a train and test dataset.
- We keep the test dataset separate for later use and split the train dataset into two parts. We developed our model in Google's Colab environment.

2.4 Validation methods:

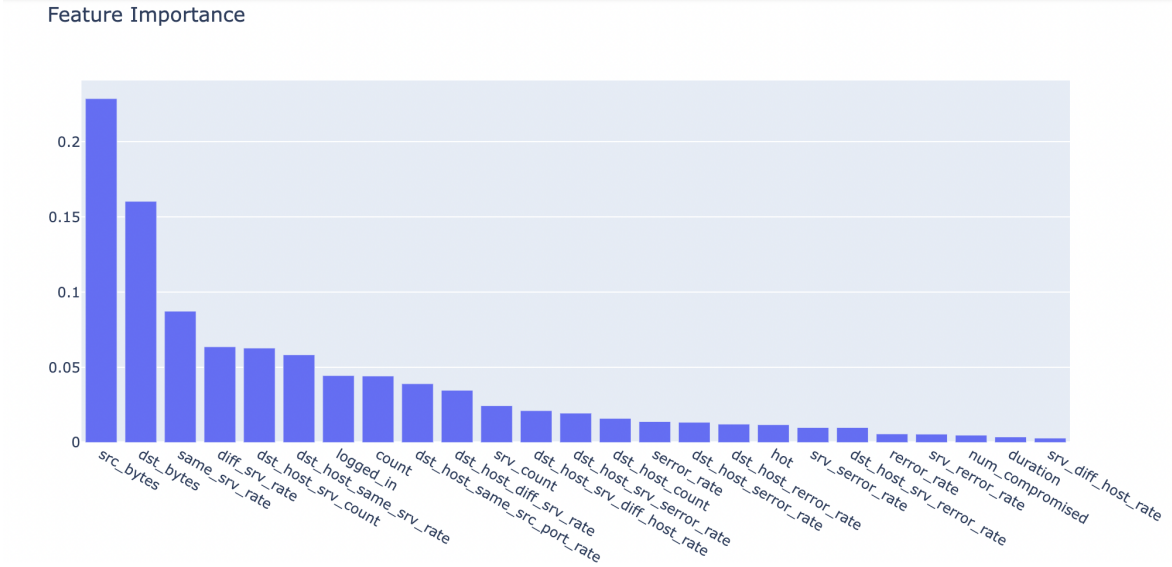
We used the K-Fold validation method to select our hyperparameters for the model. During the entire process, each set (fold) of training and the test would be

performed exactly once. It aids in the avoidance of overfitting. As we know, the best performance accuracy is achieved when a model is trained using all of the data in a single session. To combat this, k-fold cross-validation aids in the development of a generalized model. We must partition the data set into three sets, Training, Testing, and Validation, to achieve this K-Fold Cross Validation. Building models and hyperparameter assessments will be aided by the Test and Train data set. The model has been validated numerous times using the value supplied as a parameter, which is referred to as K. Here we set the value of K to five.
KFold cross validation is used to measure the generalization ability of the model. We take the RandomForestClassifier as the baseline model.

2.5 Results:

Result Process:

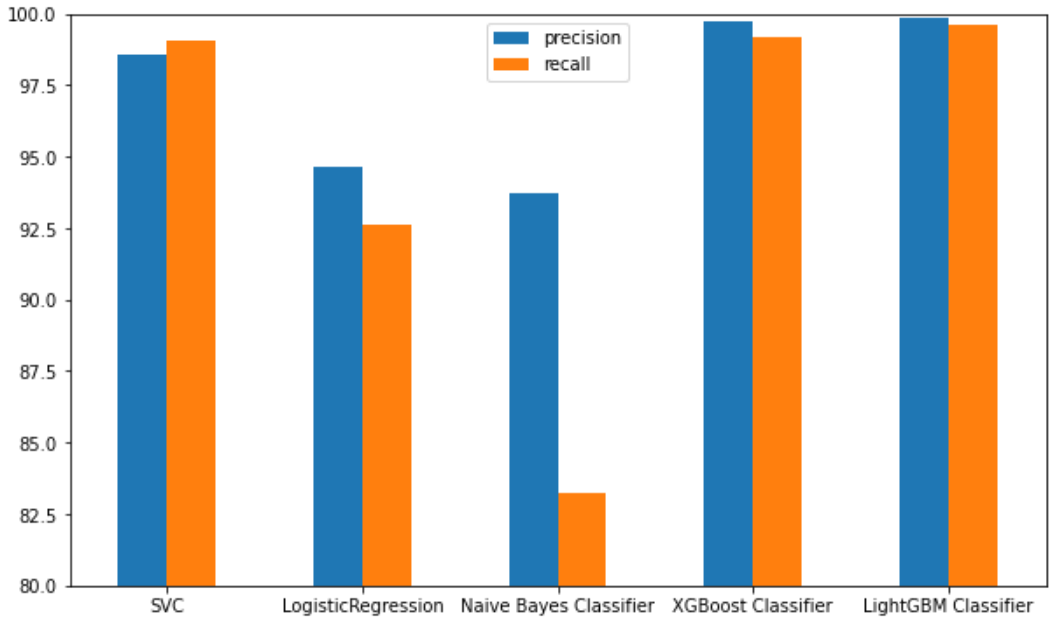
1. We decided to use the Random Forest Algorithm to find the most useful features in the dataset. Here is the plot for feature importance:



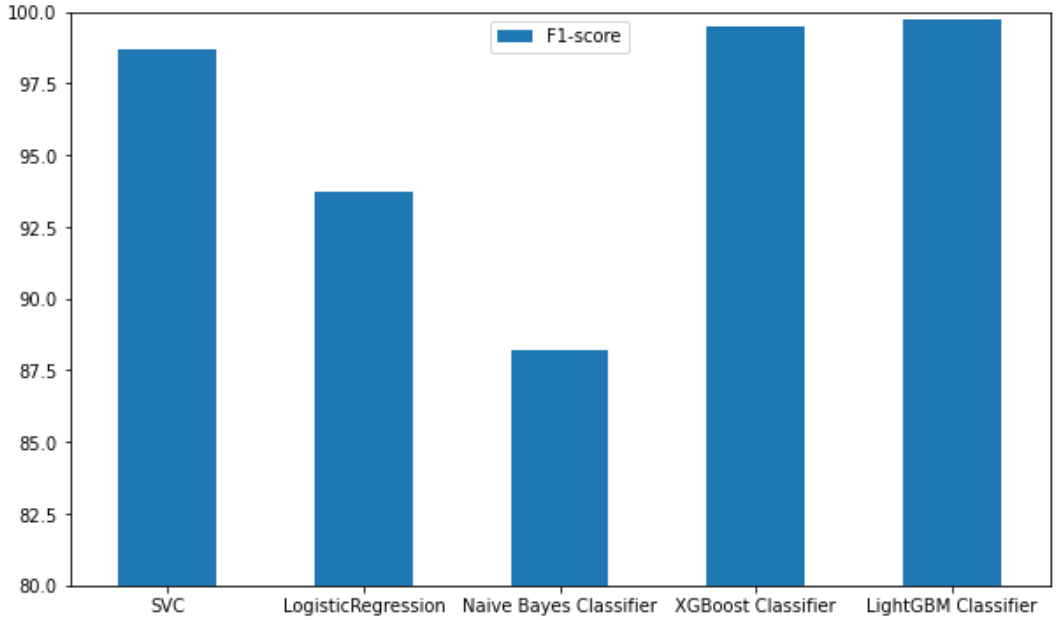
2. We had to encode various categories and features of the dataset.
3. We standardize all the numerical values in the dataset

Result & Model Selection:

1. We decided to use 5 different models on the dataset to generate the best possible precision, Recall and F1 score.



2. We used F1-Score as our metric on the test set as it serves as the harmonic mean of precision and recall values for a classification problem. It is used when we are looking for both recall and precision scores for our problem.
3. Here the results of the F1 score of different models:



4. XGBoost and LightGM have the highest F1-score and are our selected models.

Limitations:

- The dataset is quite simple and hence won't require us to implement a deep learning model.

References:

- https://kdd.ics.uci.edu/databases/kddcup99/task.html
- https://www.7sec.com/blog/the-purpose-of-intrusion-detection-and-prevention-systems
- https://www.alertlogic.com/blog/what-is-a-network-ids-and-why-do-you-need-it
- https://www.kaggle.com/sampadab17/network-intrusion-detection
- https://www.securityroundtable.org/the-growing-role-of-machine-learning-in-cybersecurity/