Cyber Attack Detection in IoT Systems

CS698 T Group 13

Team Members:

1. Ayushi Mishra (21111262)
2. Shubham Sinha (21111409)
3. Rahul Kumar (21111069)

Literature Review

1. A. Abeshu and N. Chilamkurti, "Deep Learning: The Frontier for Distributed Attack Detection in Fog-to-Things Computing," in *IEEE Communications Magazine*, vol. 56, no. 2, pp. 169-175, Feb. 2018, doi: 10.1109/MCOM.2018.1700332.

2. E. Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos and P. Burnap, "A Supervised Intrusion Detection System for Smart Home IoT Devices," in *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9042-9053, Oct. 2019, doi: 10.1109/JIOT.2019.2926365.

3. N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac and P. Faruki, "Network Intrusion Detection for IoT Security Based on Learning Techniques," in *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2671-2701, thirdquarter 2019, doi: 10.1109/COMST.2019.2896380.

4. E. Benkhelifa, T. Welsh and W. Hamouda, "A Critical Review of Practices and Challenges in Intrusion Detection Systems for IoT: Toward Universal and Resilient Systems," in *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3496-3509, Fourthquarter 2018, doi: 10.1109/COMST.2018.2844742.

5. M. Roopak, G. Yun Tian and J. Chambers, "Deep Learning Models for Cyber Security in IoT Networks," *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, 2019, pp. 0452-0457, doi: 10.1109/CCWC.2019.8666588.

6. Bandekar, Ashutosh & Javaid, Ahmad. (2017). Cyber-attack Mitigation and Impact Analysis for Low-power IoT Devices. 1631-1636. 10.1109/CYBER.2017.8446380.

Authors and Year	Dataset	Classification Type	Mechanism	Evaluation Metrics
Pahl et al. 2018	Own	Multiclass	K-Means and Birch Clustering	Acc=96.3%
Diro et al. 2018	NSL-KDD	Multiclass	Neural Network	Acc= 98.27
Anthi et al. 2018	Own	Binary	Naïve Bayes	N/A
D'angelo et al. 2015	NSL-KDD	Binary	U-BRAIN	Acc=94.1
Pajouh et al. 2018	NSL- KDD	Two-Tier	Naïve Bayes	I. R = 84.82
Milon et al. 2019	DS2OS Traffic Traces	Multiclass	ANN, SVM, DT, RF, Logistic Regression	Acc = 99.4 % for RF
Our System	DS2OS Traffic Traces	Multiclass + Handled the class imbalance problem (Smote Algorithm)	XGBoost , LightGBM, Dense Neural Network, RNN, LSTM, GRU, CNN + ML Algorithms(SVM, DT, RF)	Acc = **98.81** % for **GRU/Deep NN** Acc = **98.8% for CNN** **CNN + LSTM = 98.69%** Acc = **99.27** % for **LightGBM**

Dataset Collection

1. A virtual IoT environment is created using Distributed Smart Space Orchestration System (DS2OS). Separate systems, software, and sensors are combined into a unified management platform using IoT orchestration.

Light Controller

Motion Sensor

Sources

Washing Service

Thermostat

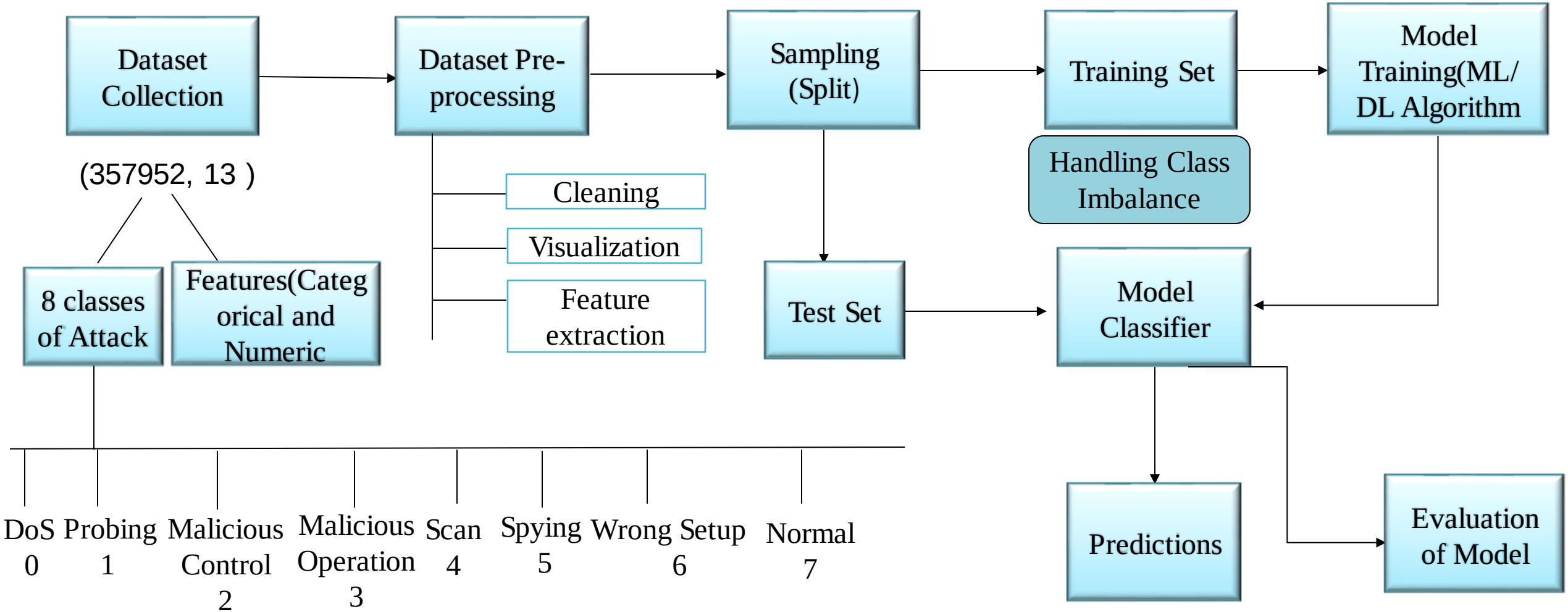
Door Lock Service

Smartphone

Smart Home Environment

Rooms, Kitchen, Garage, Entrance, Bathroom, Living Room, Dining Room

Proposed Method



Dataset

1. The dataset was collected from Kaggle.
2. A virtual IoT smart home environment was created that produced synthetic data. All the services communicated with each other using MQTT protocol.
3. The dataset contains 357,952 samples and 13 features.
4. The dataset has 347,935 Normal data and 10,017 anomalous data and contains eight classes which were classified.
5. There are seven classes of attacks:

    - **Denial of Service** (Dos) : Having too many traffic in a single source and receiver.
    - **Data Type Probing**: a malicious node can write a different datatype than an intended datatype.
    - **Malicious Control:** attacker can gain a session key and can control the entire system.
    - **Malicious Operation**: it is generally caused by the malware which affects the device's performance.
    - **Scan**: when data is acquired by hardware while scanning, the data may get corrupted sometimes.
    - **Spying**: the attacker uses a backdoor system to cause vulnerabilities in the system and steal confidential information.
    - **Wrong Setup**: a wrong system setup can also disrupt the data.

Classes of Attack	Count
Denial of Service	5780
Data Type Probing	342
Malicious Control	889
Malicious Operation	805
Scan	1547
Spying	532
Wrong Setup	122

Features	Data Type
Source ID	Categorical
Source Address	Categorical
Source Location	Categorical
Source Destination	Categorical
Destination Service Address	Categorical
Destination Service Type	Categorical
Destination Location	Categorical
Accessed Node Address	Categorical
Accessed Node Type	Categorical
Operation	Categorical
Value	Continuous
Timestamp	Discrete
Normality	Categorical

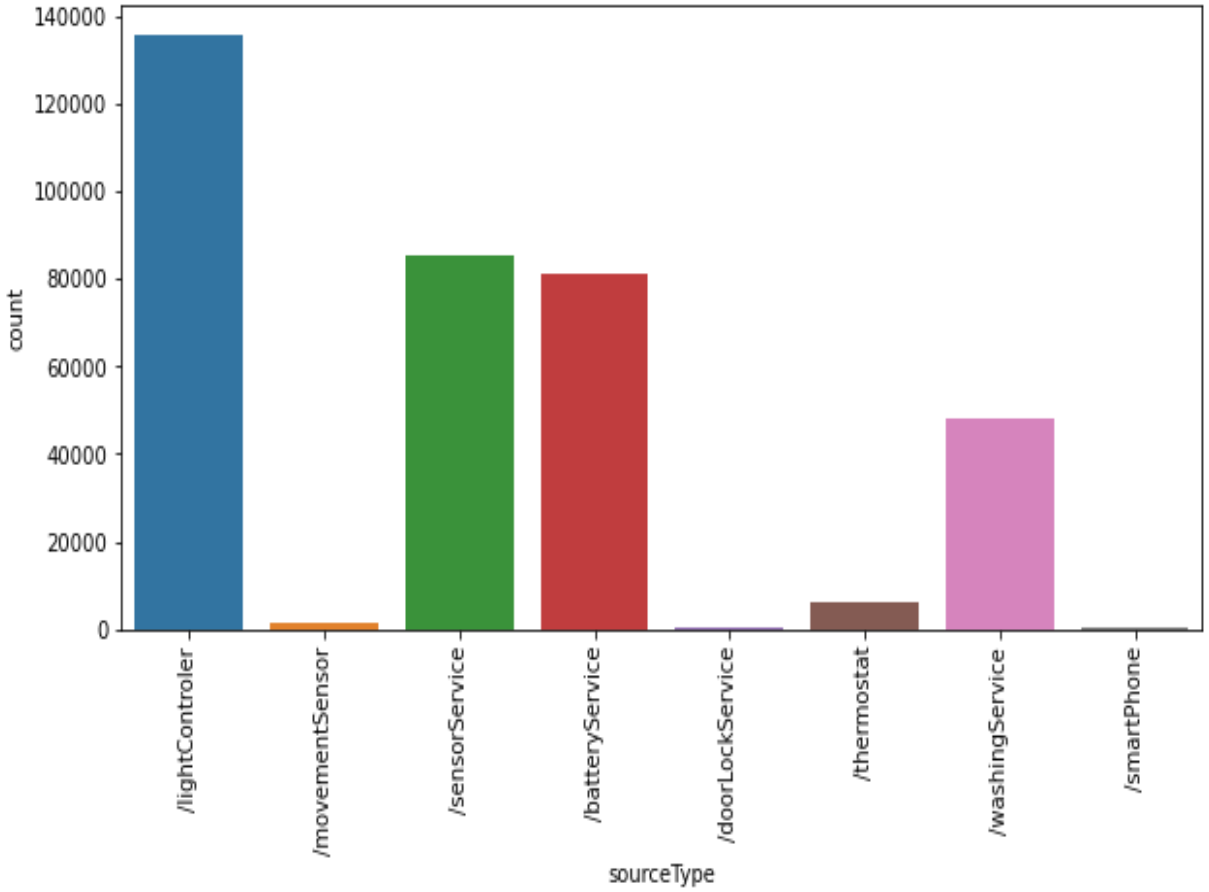
Data Preprocessing

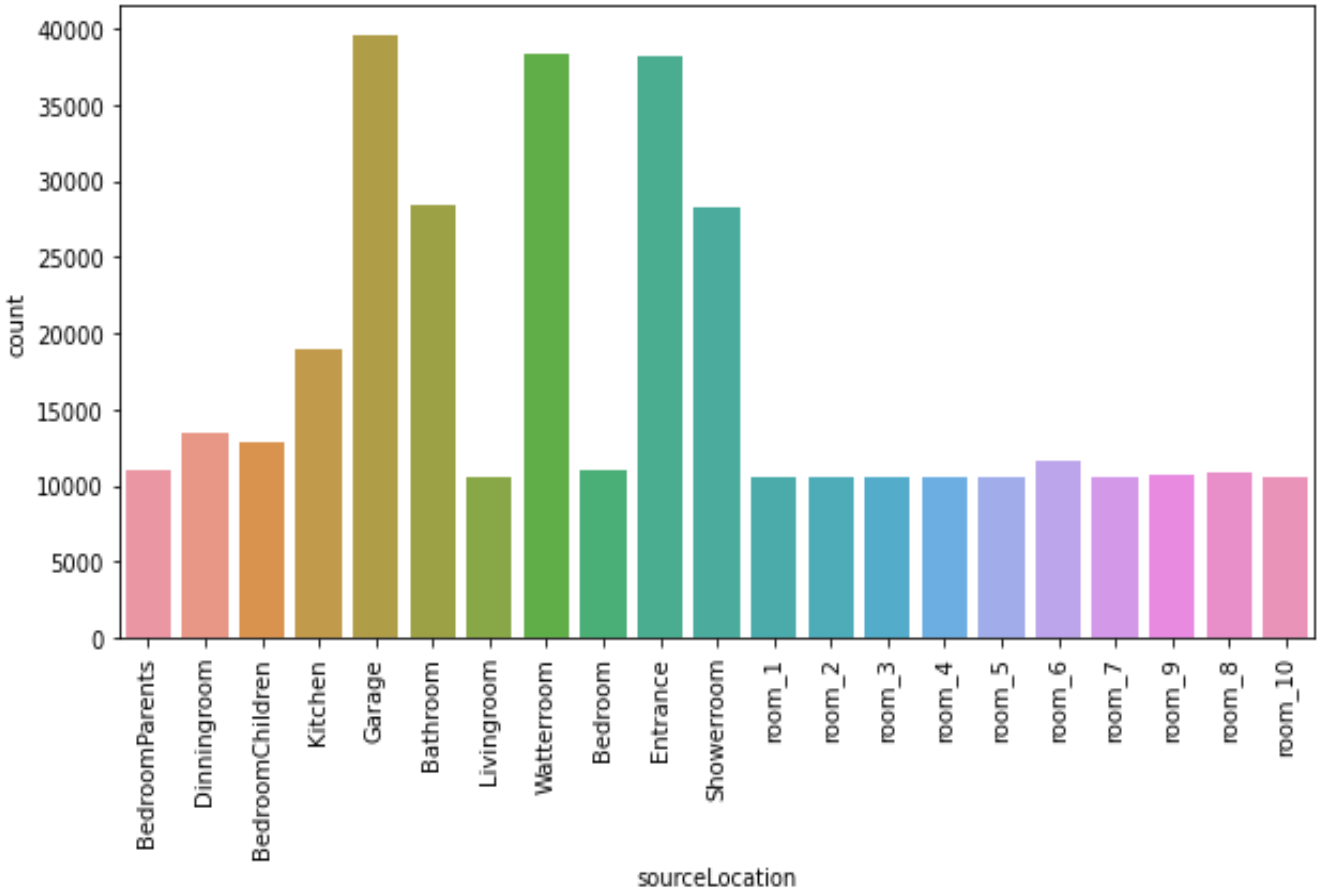
1. **Data Cleaning:**
    - Handling Missing Values: Features like Accessed Node Type and Value contain some empty fields. The empty values for Accessed Node Type are dropped and the empty fields in Value are filled with mean values.

2. Features like Timestamp was dropped as it does not have any significant impact on the data analysis.

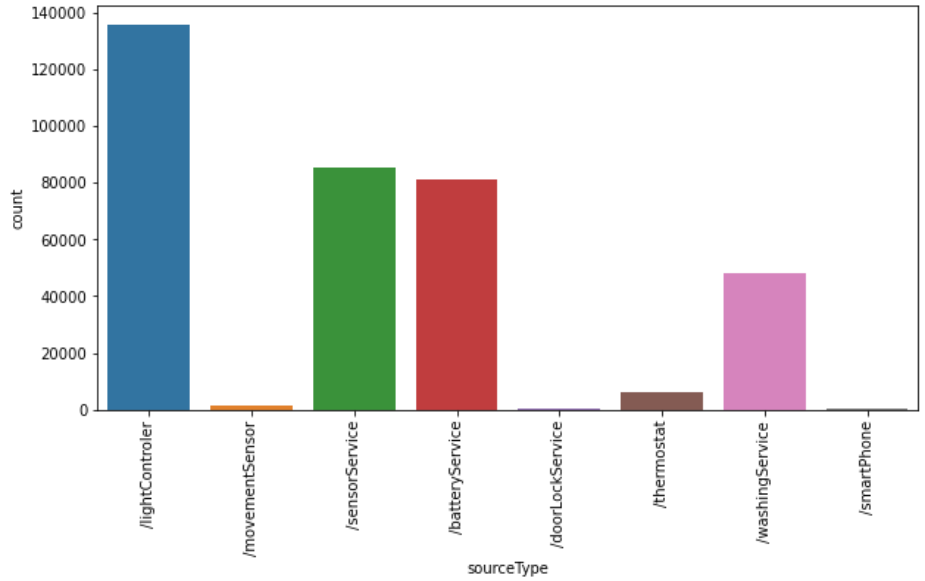
3. **Feature Engineering:**
    - It is vital to look at the datatype of all the features.
    - The categorical features are converted into vectors. We have used Label Encoding to convert it into feature vectors.
    - Using Label Encoding will not increase the dimensionality of the data and processing time will be less.
    - Normalizing the Training data to make it in a same scale.

Data Analysis

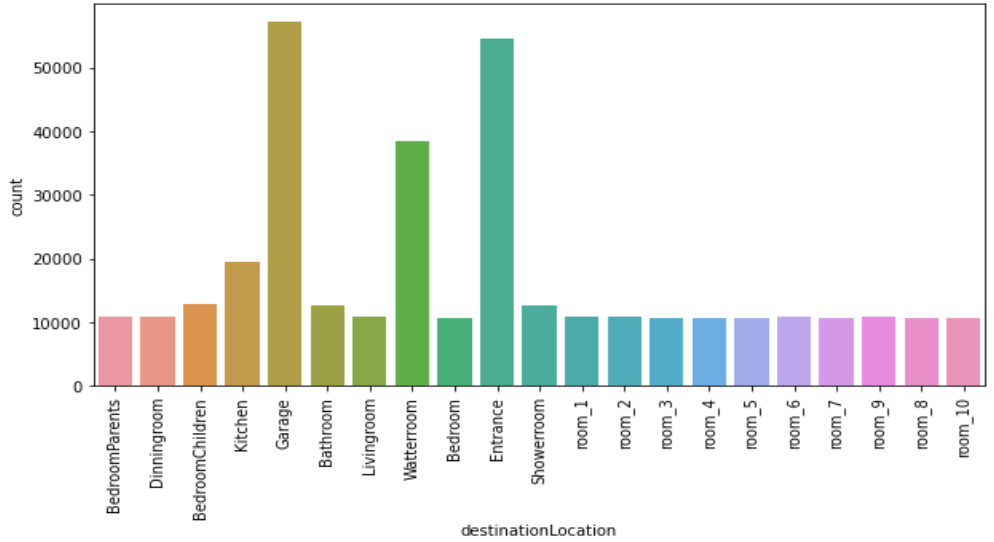




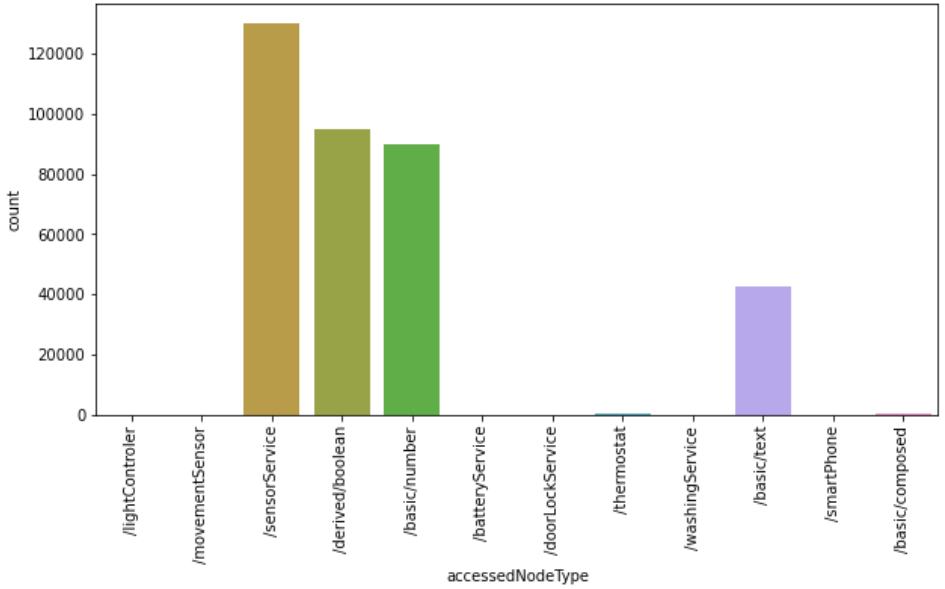
Feature Description



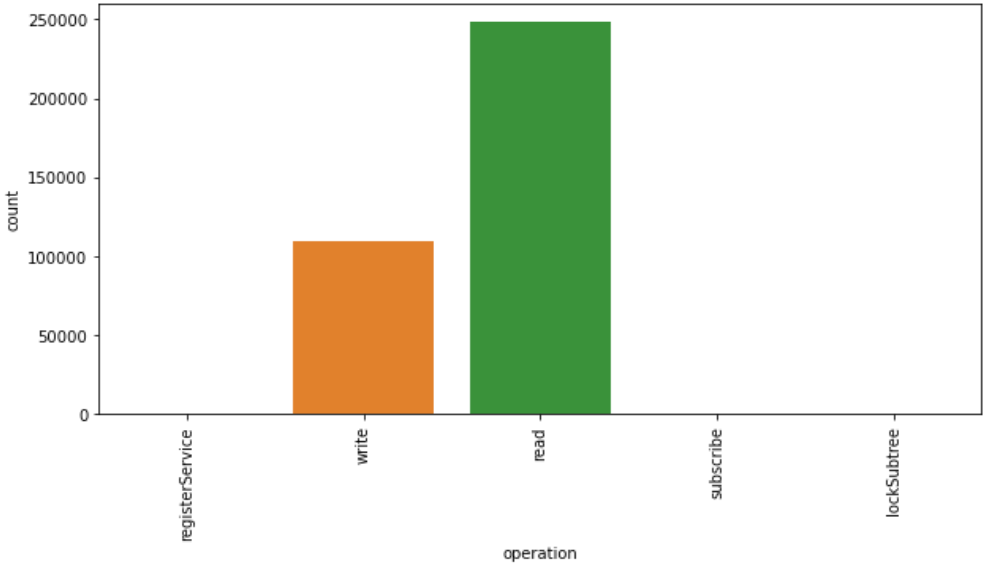
I) Source Type



II) Destination Location

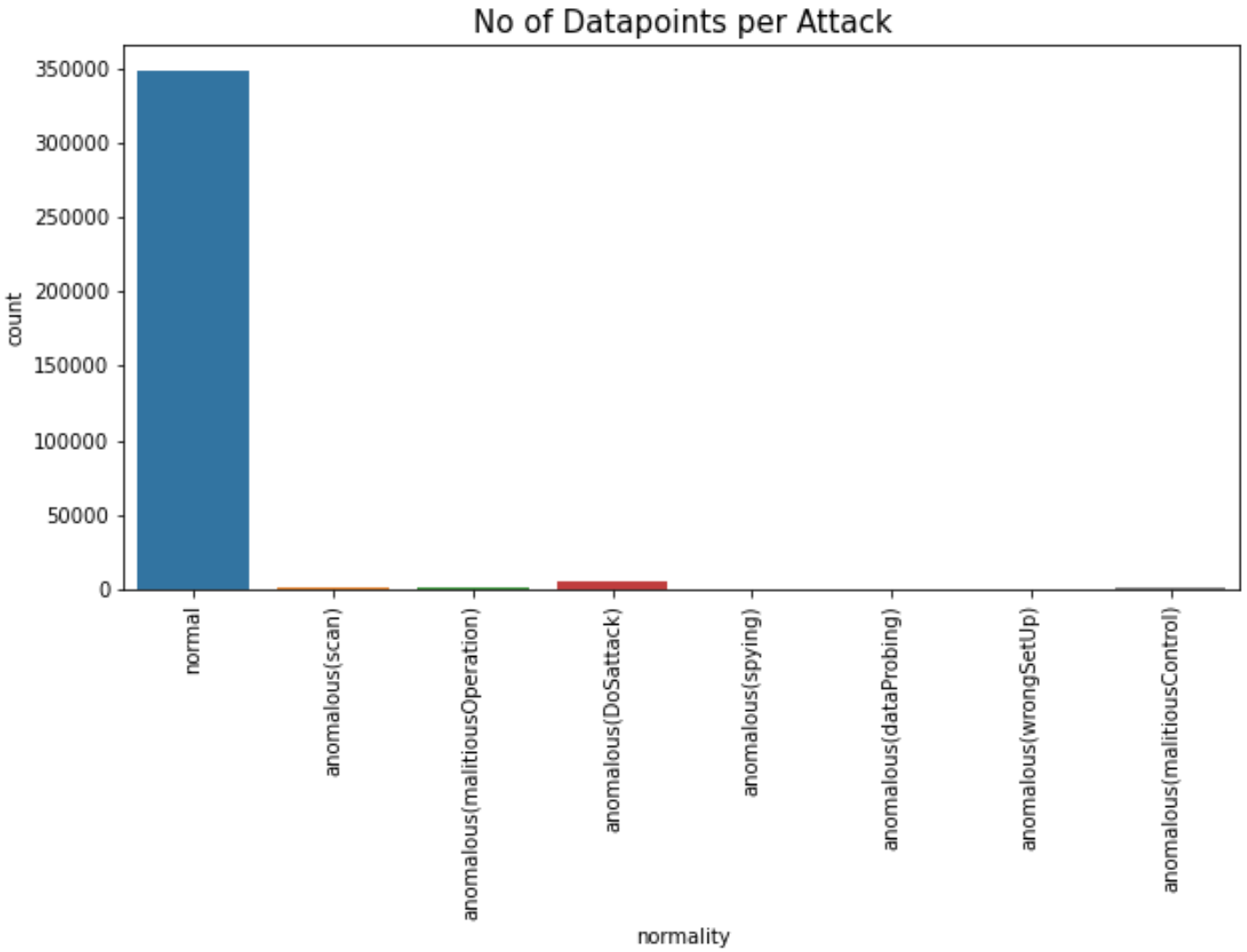


III) Accessed Node Type



IV) Destination

Labels Description



Labels	Total % of the data
Anomalous(DoSattack)	1.61%
Anomalous(scan	0.43%
Anomalous(malitiousControl)	0.24%
Anomalous(malitiousOperation)	0.22%
Anomalous(spying)	0.14%
Anomalous(dataProbing)	0.09%
Anomalous(wrongSetUp)	0.03%

Class Imbalance Problem

Training Models before handling Class Imbalance Problem

1. Considered some traditional Machine Learning classification algorithms like Logistic Regression, Decision Trees and Random Forest.

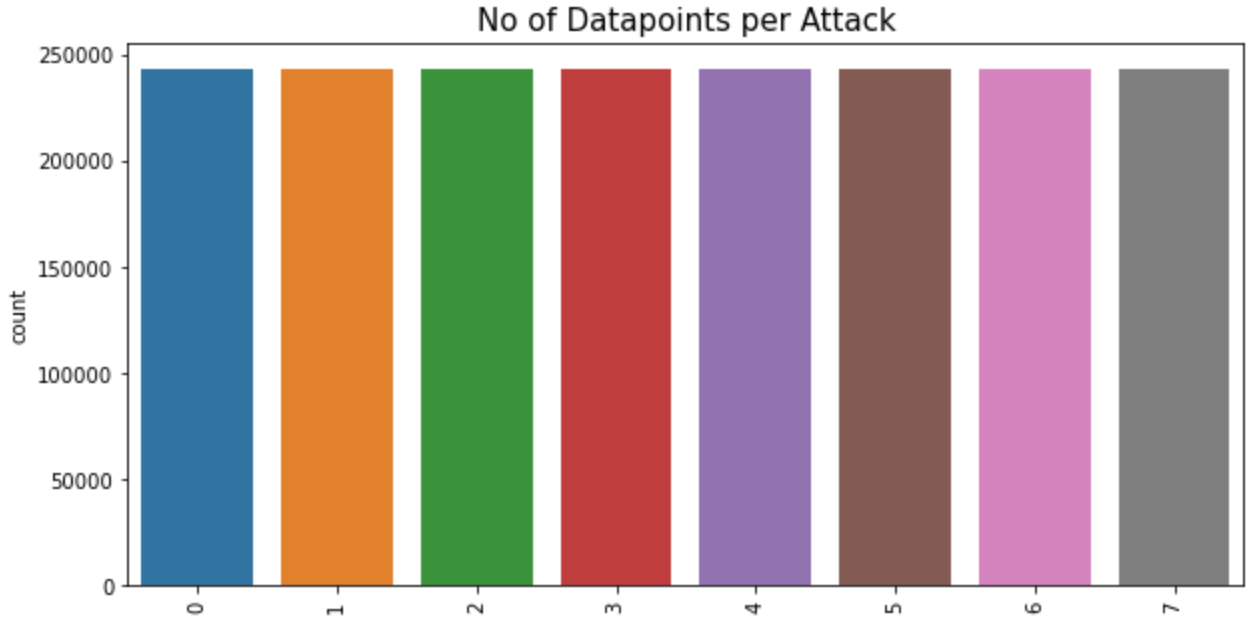
ML Algorithms	Accuracy	Test Time (Micro Secs)
Logistic Regression	98.38%	0.03
Decision Trees	99.40%	0.04
Random Forest	99.40%	0.78

2. To avoid the problem of overfitting, Class Imbalance problem needs to be handled.

1. Handling Class Imbalance Problem: **Using SMOTE Algorithm**

Before OverSampling, counts of label '0': 4045
Before OverSampling, counts of label '1': 237
Before OverSampling, counts of label '2': 618
Before OverSampling, counts of label '3': 567
Before OverSampling, counts of label '4': 1102
Before OverSampling, counts of label '5': 370
Before OverSampling, counts of label '6': 93
Before OverSampling, counts of label '7': 243534

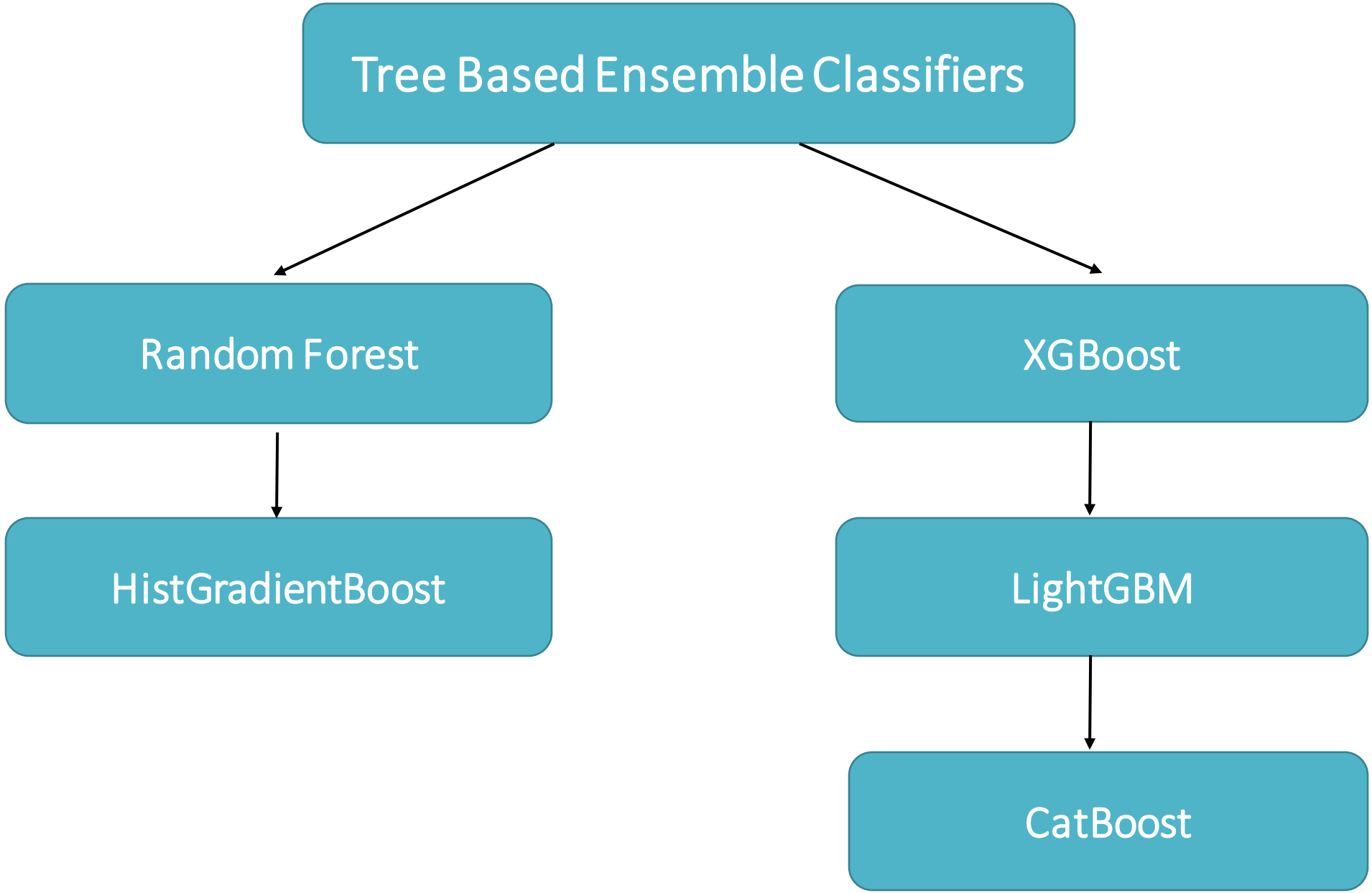
After OverSampling, counts of label '0': 243534
After OverSampling, counts of label '1': 243534
After OverSampling, counts of label '2': 243534
After OverSampling, counts of label '3': 243534
After OverSampling, counts of label '4': 243534
After OverSampling, counts of label '5': 243534
After OverSampling, counts of label '6': 243534
After OverSampling, counts of label '7': 243534



Machine Learning Models

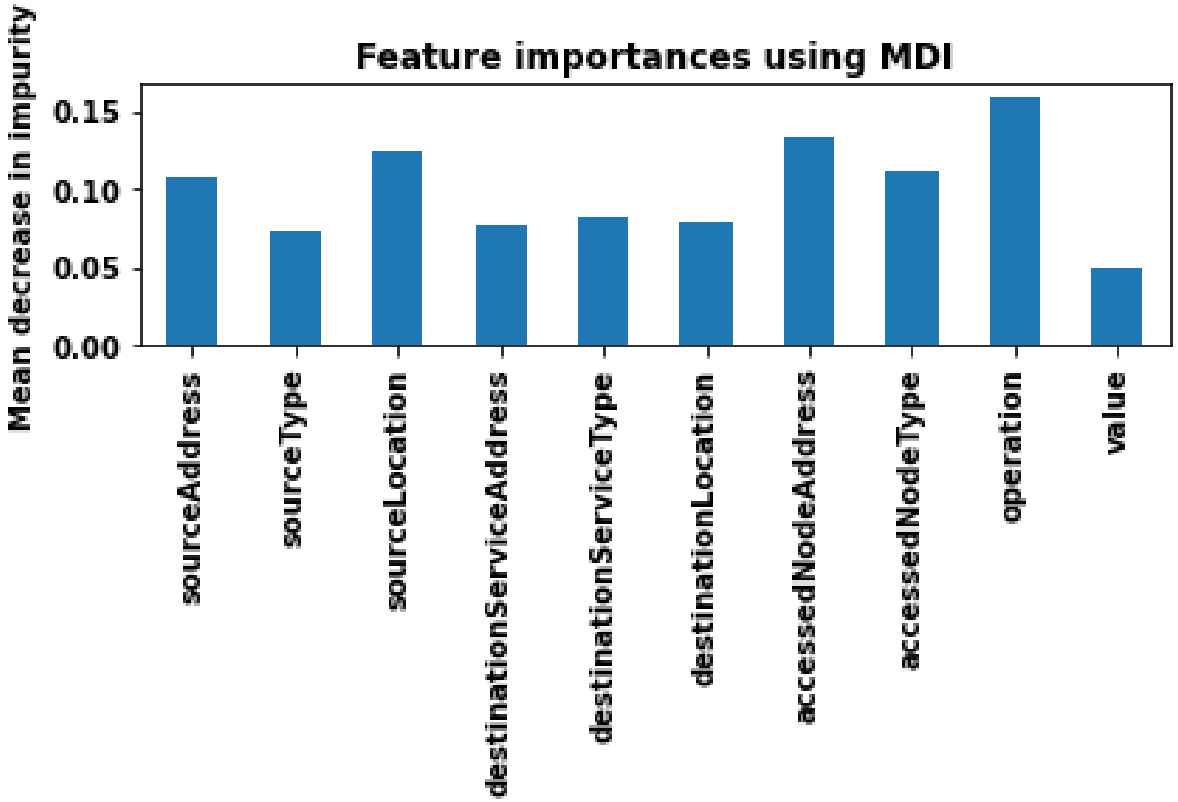
1. First we trained our model with traditional ML algorithms.
2. Logistic Regression performs poorly in the model prediction with the accuracy of 41%.
3. Decision Trees gave the accuracy of 74.5%
4. Random Forest gave the accuracy of 98.80%.

ML Algorithms	Accuracy	Test Time (Micro Secs)
Logistic Regression	41%	0.32
Decision Trees	74.5%	0.062
Random Forest	98.80	34


Tree Based Ensemble Classifiers
Random Forest
HistGradientBoost
XGBoost
LightGBM
CatBoost

Reasons for Tree based Ensemble Classifiers

1. Non-linear classifier
2. Unbounded by kernel
3. Structured dataset
4. Less training time
5. Resistance to overfitting
6. Hardware optimizations like cache friendliness, parallel jobs.
7. Dataset friendliness like support for missing values, categorical values.




```
0 anomalous(DoSattack)
1 anomalous(dataProbing)
2 anomalous(malitiousControl)
3 anomalous(malitiousOperation)
4 anomalous(scan)
5 anomalous(spying)
6 anomalous(wrongSetUp)
7 normal
```

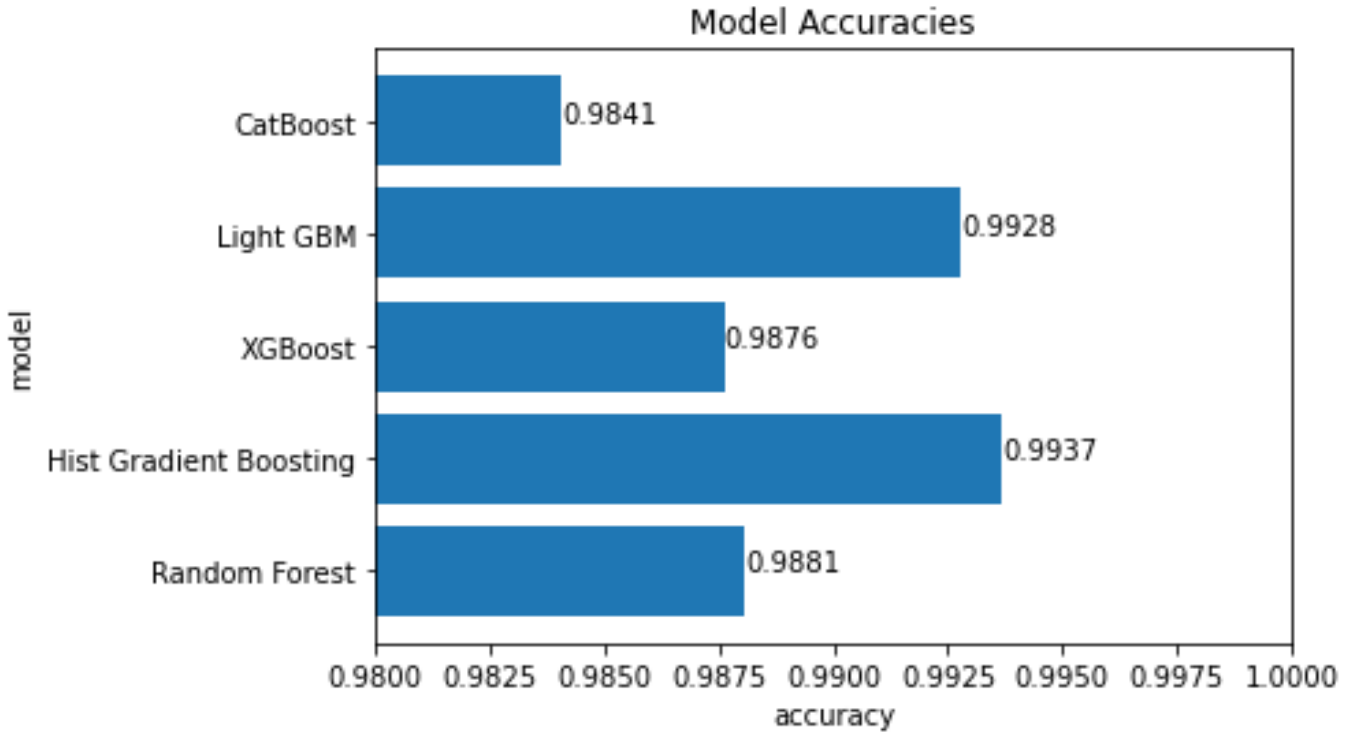
```
cm = confusion_matrix(y_test, predictions)
print(cm)
```

```
[[  1735      0      0      0      0      0      0      0]
 [     0    105      0      0      0      0      0      0]
 [     0      0    271      0      0      0      0      0]
 [     0      0      0    238      0      0      0      0]
 [     0      0      0      0    445      0      0      0]
 [     0      0      0      0      0    162      0      0]
 [     0      0      0      0      0      0     29      0]
 [   747      0      1     26      0      3      0 103624]]
```

	precision	recall	f1-score	support
0	0.699033	1.000000	0.822860	1735
1	1.000000	1.000000	1.000000	105
2	0.996324	1.000000	0.998158	271
3	0.901515	1.000000	0.948207	238
4	1.000000	1.000000	1.000000	445
5	0.981818	1.000000	0.990826	162
6	1.000000	1.000000	1.000000	29
7	1.000000	0.992558	0.996265	104401
accuracy			0.992764	107386
macro avg	0.947336	0.999070	0.969539	107386
weighted avg	0.994882	0.992764	0.993373	107386

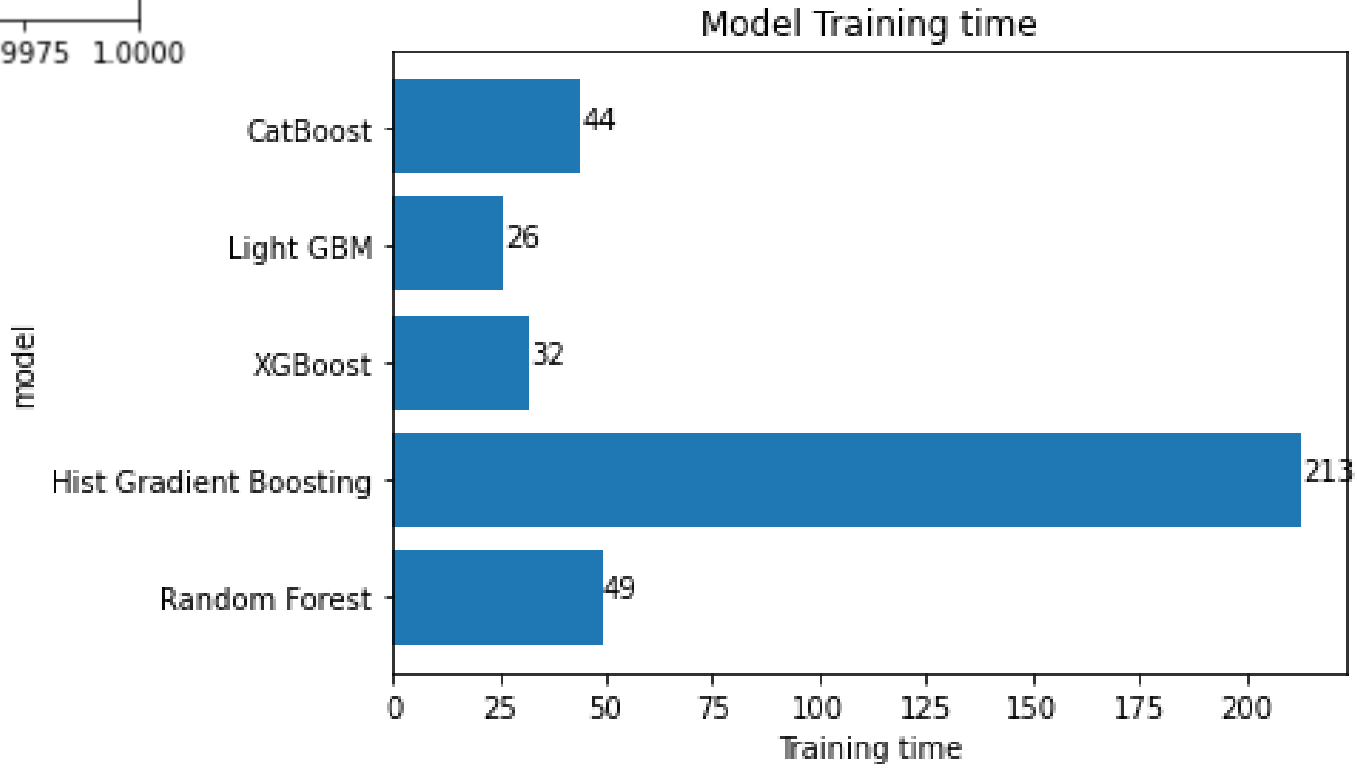
Trained Hyper-parameters

Function	XGBoost	CatBoost	Light GBM
Important parameters which control overfitting	1. **learning_rate or eta** – optimal values lie between 0.01-0.2 2. **max_depth** 3. **min_child_weight:** similar to min_child leaf; default is 1	1. **Learning_rate** 2. **Depth** - value can be any integer up to 16. Recommended - [1 to 10] 3. No such feature like min_child_weight 4. **l2-leaf-reg**: L2 regularization coefficient. Used for leaf value calculation (any positive integer allowed)	1. **learning_rate** 2. **max_depth**: default is 20. Important to note that tree still grows leaf-wise. Hence it is important to tune **num_leaves** (number of leaves in a tree) which should be smaller than 2^(max_depth). It is a very important parameter for LGBM 3. **min_data_in_leaf**: default=20, alias= min_data, min_child_samples
Parameters for controlling speed	1. **colsample_bytree**: subsample ratio of columns 2. **subsample**: subsample ratio of the training instance 3. **n_estimators**: maximum number of decision trees; high value can lead to overfitting	1. **rsm:** Random subspace method. The percentage of features to use at each split selection 2. No such parameter to subset data 3. **iterations**: maximum number of trees that can be built; high value can lead to overfitting	1. **feature_fraction:** fraction of features to be taken for each iteration 2. **bagging_fraction:** data to be used for each iteration and is generally used to speed up the training and avoid overfitting 3. **num_iterations:** number of boosting iterations to be performed; default=100



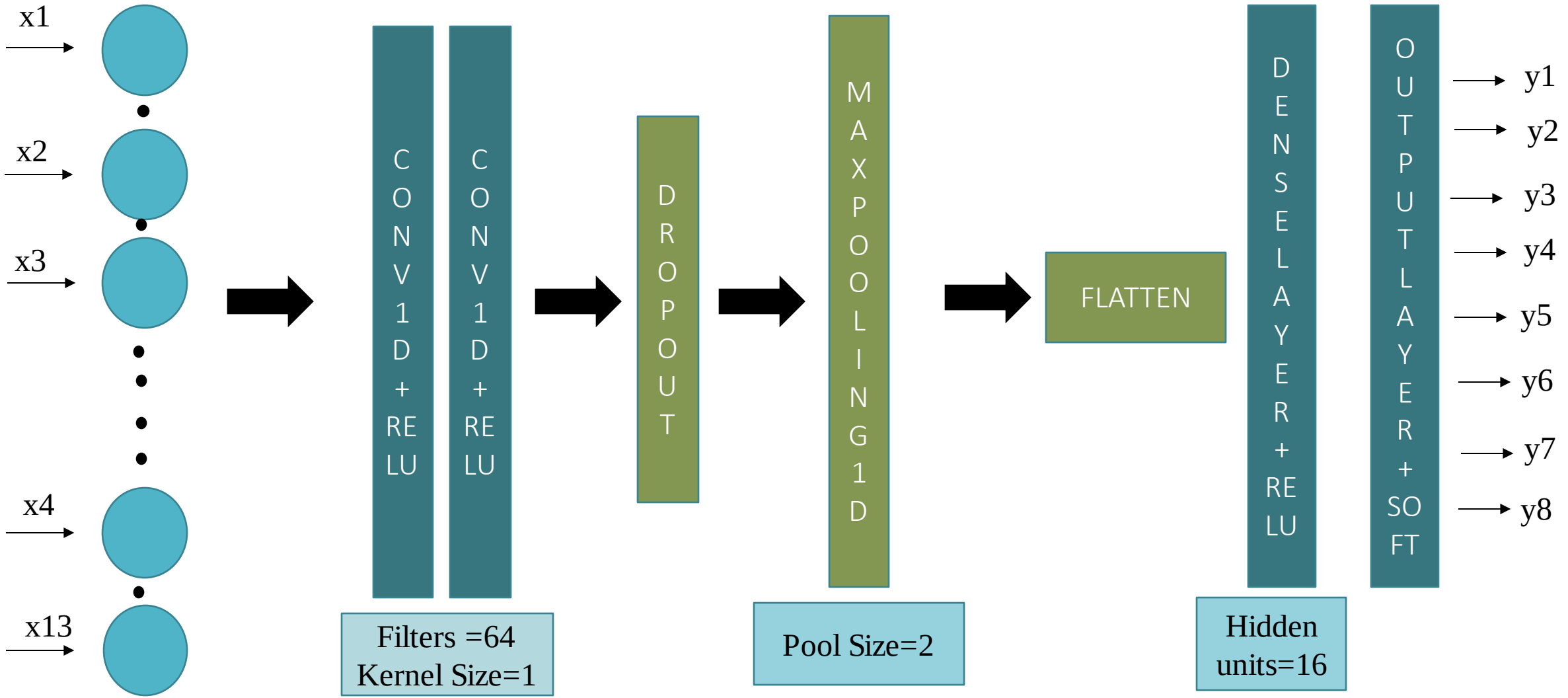
Searched Parameters	Values
max_depth, depth	5
learning_rate	0.1,0.25
iterations	25
Feature_fraction, subsample	0.5, 0.6

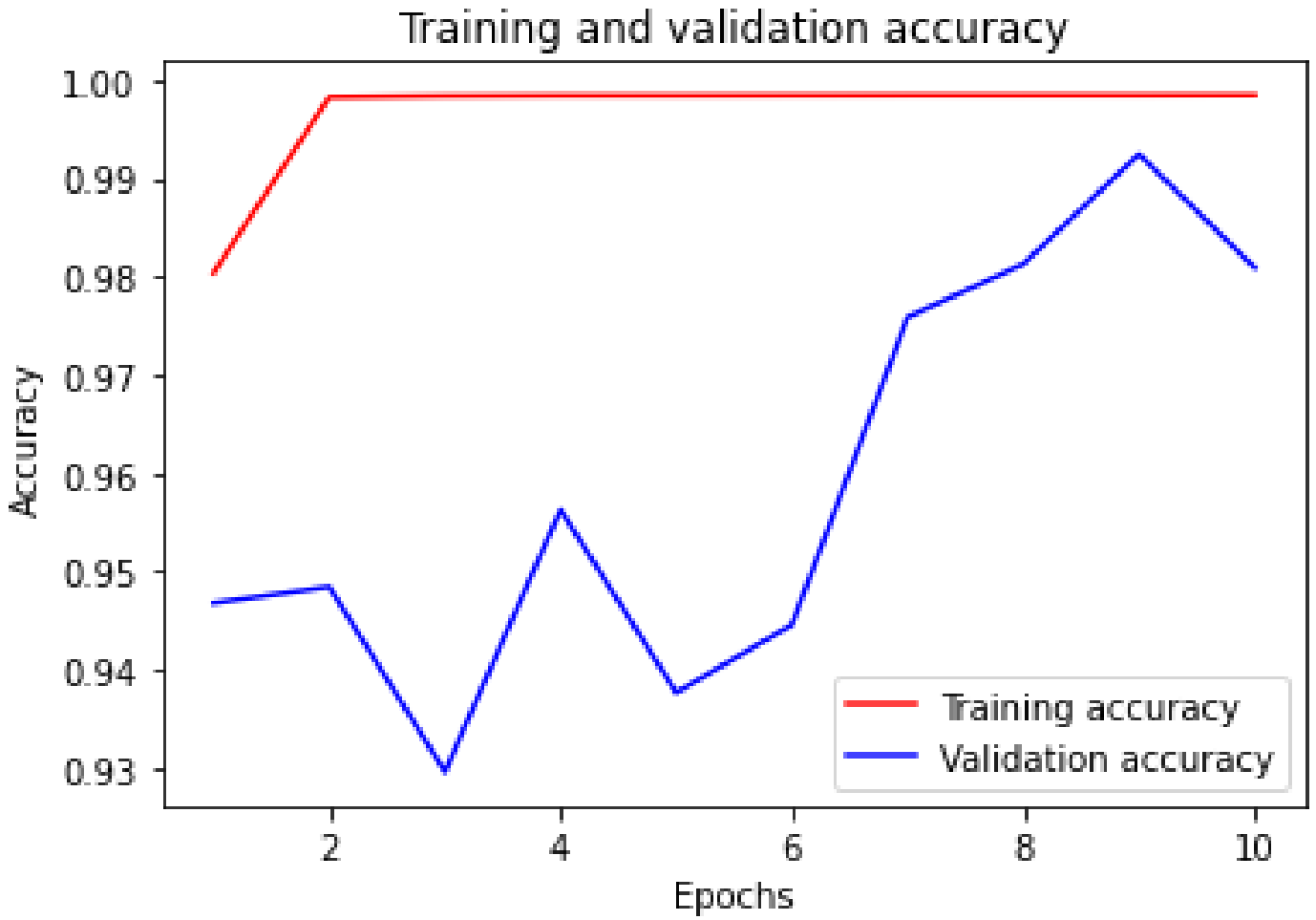
Fixed Parameters	Values
objective	Multiclass, multi:softmax
eval_metric	Mlogloss, multi_logloss
num_class	8



Deep Learning Models

1D Convolutional Neural Network (1D CNN)





Epoch	10
Batch Size	512
Loss	Categorical Crossentropy
Activation Function	ReLU & Softmax
Optimizer	Adam
Accuracy	98.8%

DL Models

Layer (type)	Output Shape	Param #
simple_rnn (SimpleRNN)	(None, 1, 8)	160
dropout (Dropout)	(None, 1, 8)	0
simple_rnn_1 (SimpleRNN)	(None, 8)	136
dropout_1 (Dropout)	(None, 8)	0
dense (Dense)	(None, 8)	72
activation (Activation)	(None, 8)	0

Total params: 368
Trainable params: 368
Non-trainable params: 0

Simple RNN

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 1, 8)	640
dropout_2 (Dropout)	(None, 1, 8)	0
lstm_1 (LSTM)	(None, 8)	544
dropout_3 (Dropout)	(None, 8)	0
dense_1 (Dense)	(None, 8)	72
activation_1 (Activation)	(None, 8)	0

Total params: 1,256
Trainable params: 1,256
Non-trainable params: 0

LSTM

Layer (type)	Output Shape	Param #
gru (GRU)	(None, 1, 32)	4320
dropout_4 (Dropout)	(None, 1, 32)	0
gru_1 (GRU)	(None, 1, 32)	6336
dropout_5 (Dropout)	(None, 1, 32)	0
gru_2 (GRU)	(None, 1, 32)	6336
dropout_6 (Dropout)	(None, 1, 32)	0
gru_3 (GRU)	(None, 32)	6336
dropout_7 (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 8)	264
activation_2 (Activation)	(None, 8)	0

Total params: 23,592
Trainable params: 23,592
Non-trainable params: 0

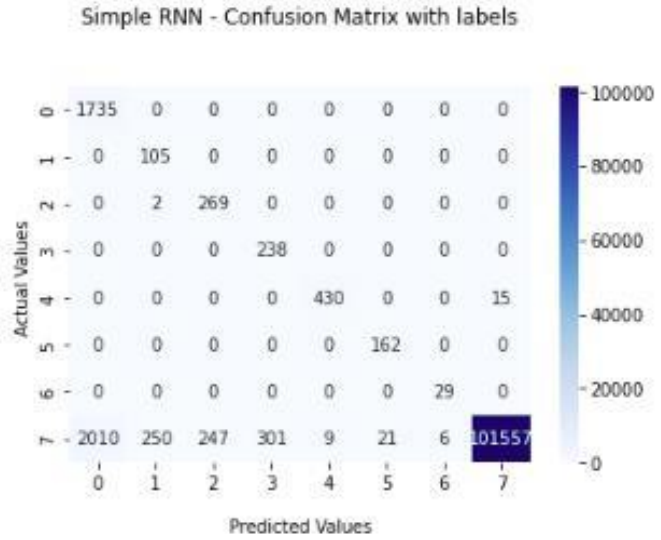
GRU

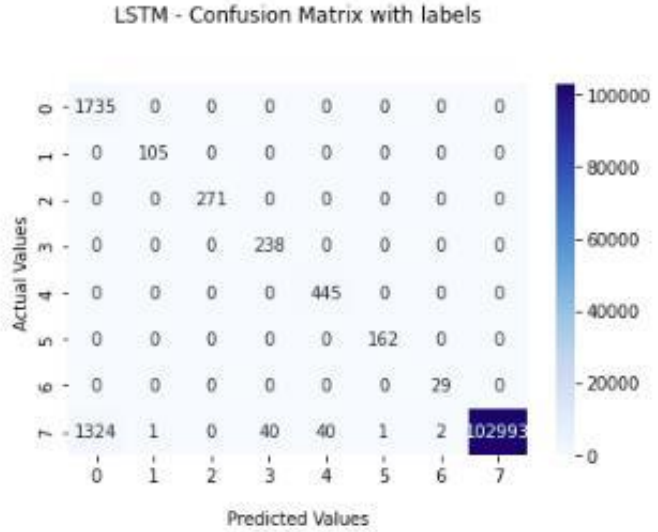
Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 11)	0
dense_3 (Dense)	(None, 1024)	12288
dropout_8 (Dropout)	(None, 1024)	0
dense_4 (Dense)	(None, 768)	787200
dropout_9 (Dropout)	(None, 768)	0
dense_5 (Dense)	(None, 8)	6152
activation_3 (Activation)	(None, 8)	0

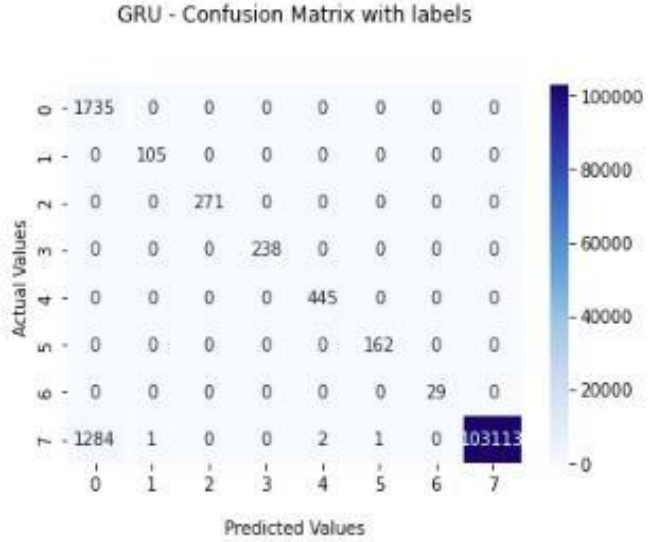
Total params: 805,640
Trainable params: 805,640
Non-trainable params: 0

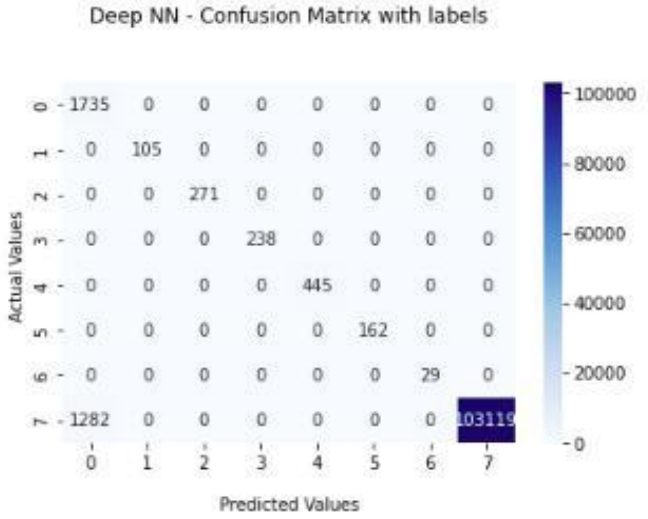
Deep NN

Results


Simple RNN - Confusion Matrix with labels
Actual Values
Predicted Values
0 1 2 3 4 5 6 7
1735 0 0 0 0 0 0 0
0 105 0 0 0 0 0 0
0 2 269 0 0 0 0 0
0 0 0 238 0 0 0 0
0 0 0 0 430 0 0 15
0 0 0 0 0 162 0 0
0 0 0 0 0 0 29 0
2010 250 247 301 9 21 6 101557
100000 80000 60000 40000 20000 0


LSTM - Confusion Matrix with labels
Actual Values
Predicted Values
0 1 2 3 4 5 6 7
1735 0 0 0 0 0 0 0
0 105 0 0 0 0 0 0
0 0 271 0 0 0 0 0
0 0 0 238 0 0 0 0
0 0 0 0 445 0 0 0
0 0 0 0 0 162 0 0
0 0 0 0 0 0 29 0
1324 1 0 40 40 1 2 102993
100000 80000 60000 40000 20000 0


GRU - Confusion Matrix with labels
Actual Values
Predicted Values
0 1 2 3 4 5 6 7
1735 0 0 0 0 0 0 0
0 105 0 0 0 0 0 0
0 0 271 0 0 0 0 0
0 0 0 238 0 0 0 0
0 0 0 0 445 0 0 0
0 0 0 0 0 162 0 0
0 0 0 0 0 0 29 0
1284 1 0 0 2 1 0 103113
100000 80000 60000 40000 20000 0


Deep NN - Confusion Matrix with labels
Actual Values
Predicted Values
0 1 2 3 4 5 6 7
1735 0 0 0 0 0 0 0
0 105 0 0 0 0 0 0
0 0 271 0 0 0 0 0
0 0 0 238 0 0 0 0
0 0 0 0 445 0 0 0
0 0 0 0 0 162 0 0
0 0 0 0 0 0 29 0
1282 0 0 0 0 0 0 103119
100000 80000 60000 40000 20000 0

Hyperparameter

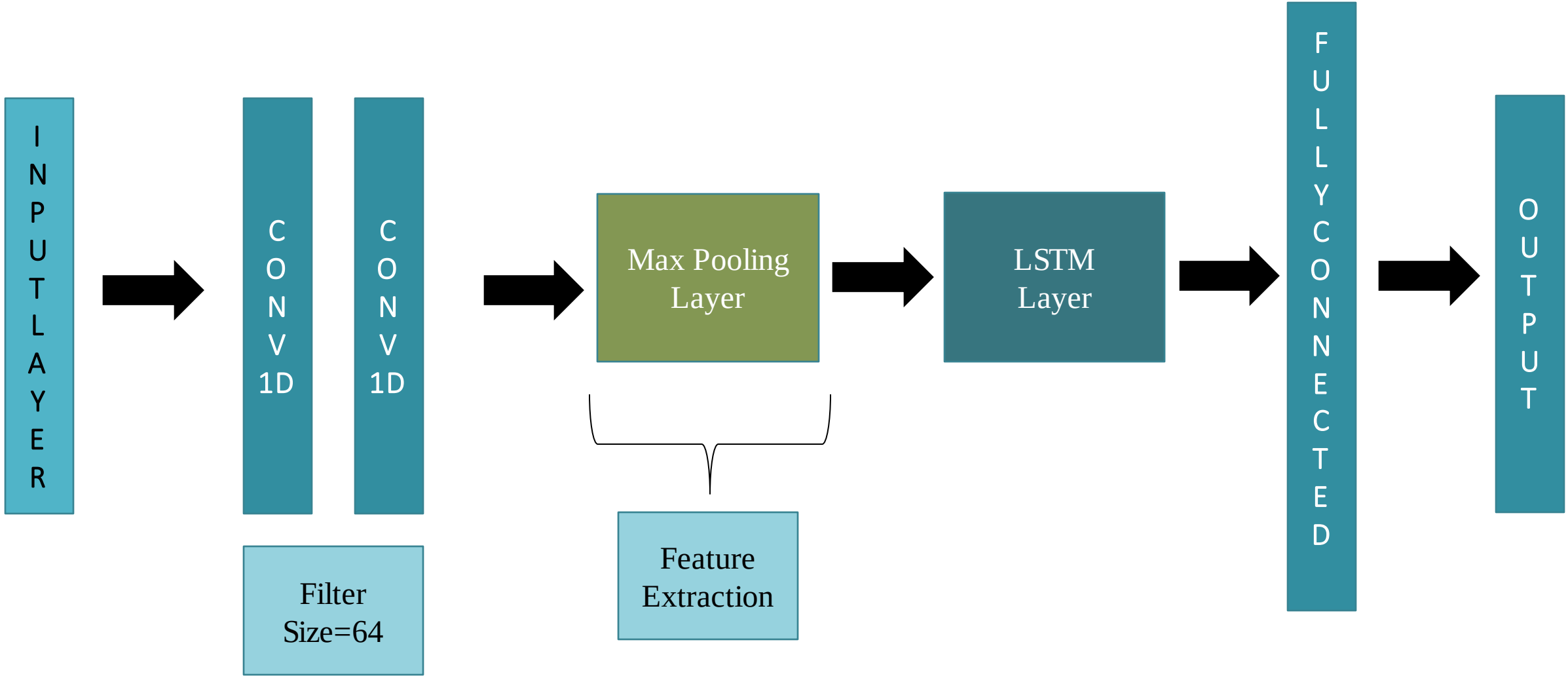
Epoch	10
Batch Size	512
Loss	Categorical Crossentropy
Activation Function	ReLU & Softmax
Optimizer	Adam

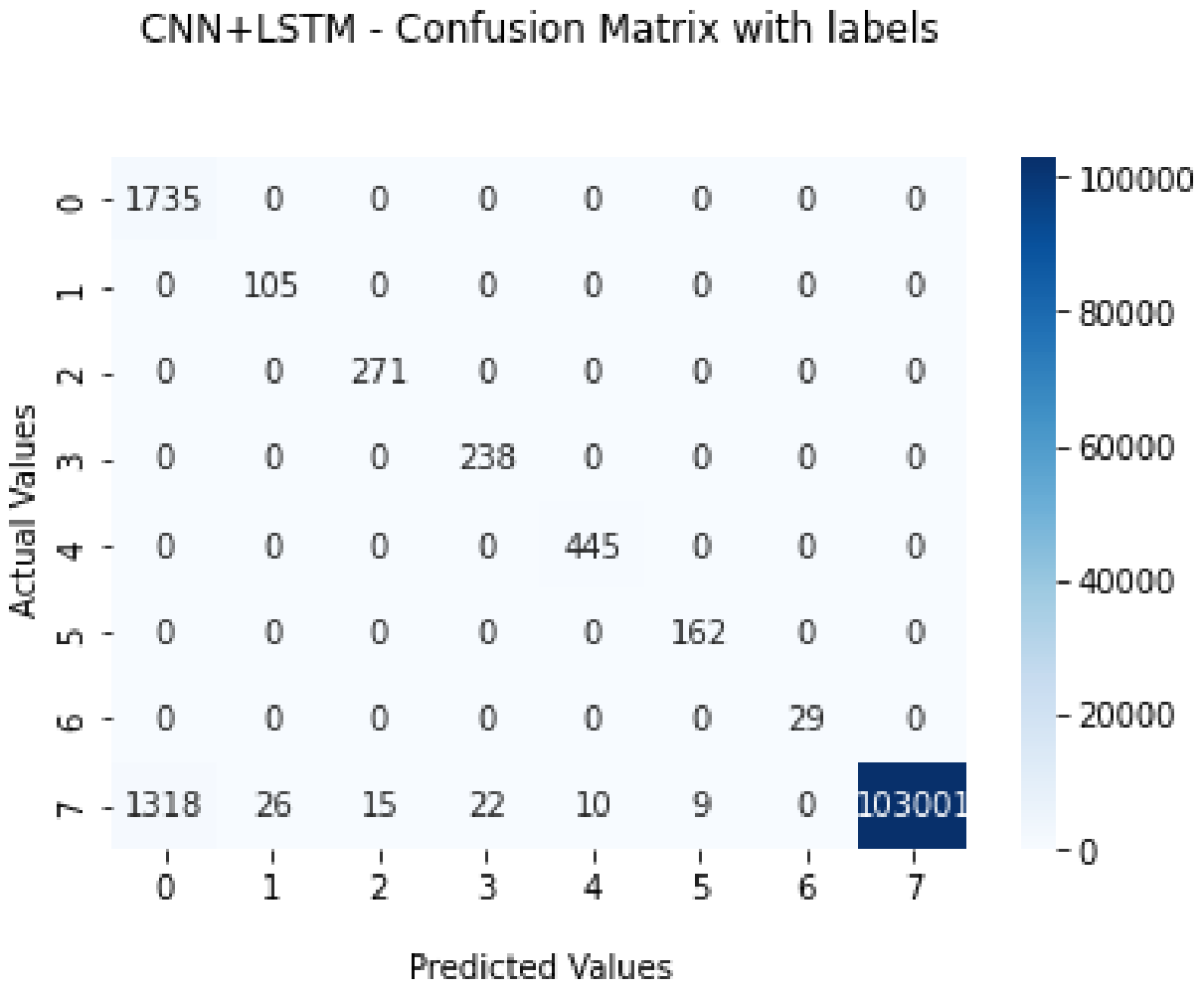
Accuracy Score

Models	Accuracy Score
Simple RNN	97.33
LSTM	98.69
GRU	98.80
Deep NN	98.81

Ensemble Methods

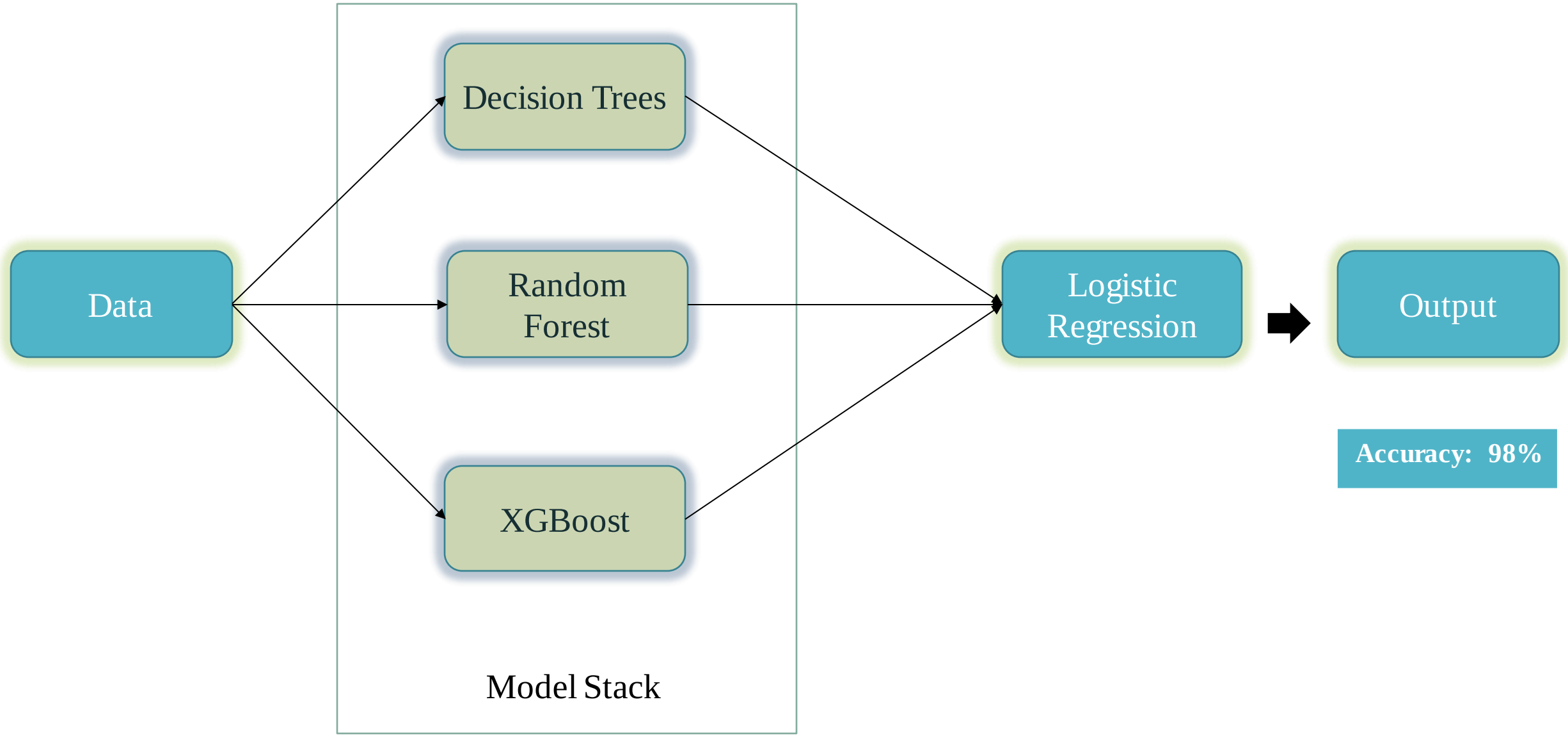
CNN + LSTM





Epoch	10
Batch Size	512
Loss	Categorical Crossentropy
Activation Function	ReLU & Softmax
Optimizer	Adam
Accuracy	98.69%

Stacked ML Method

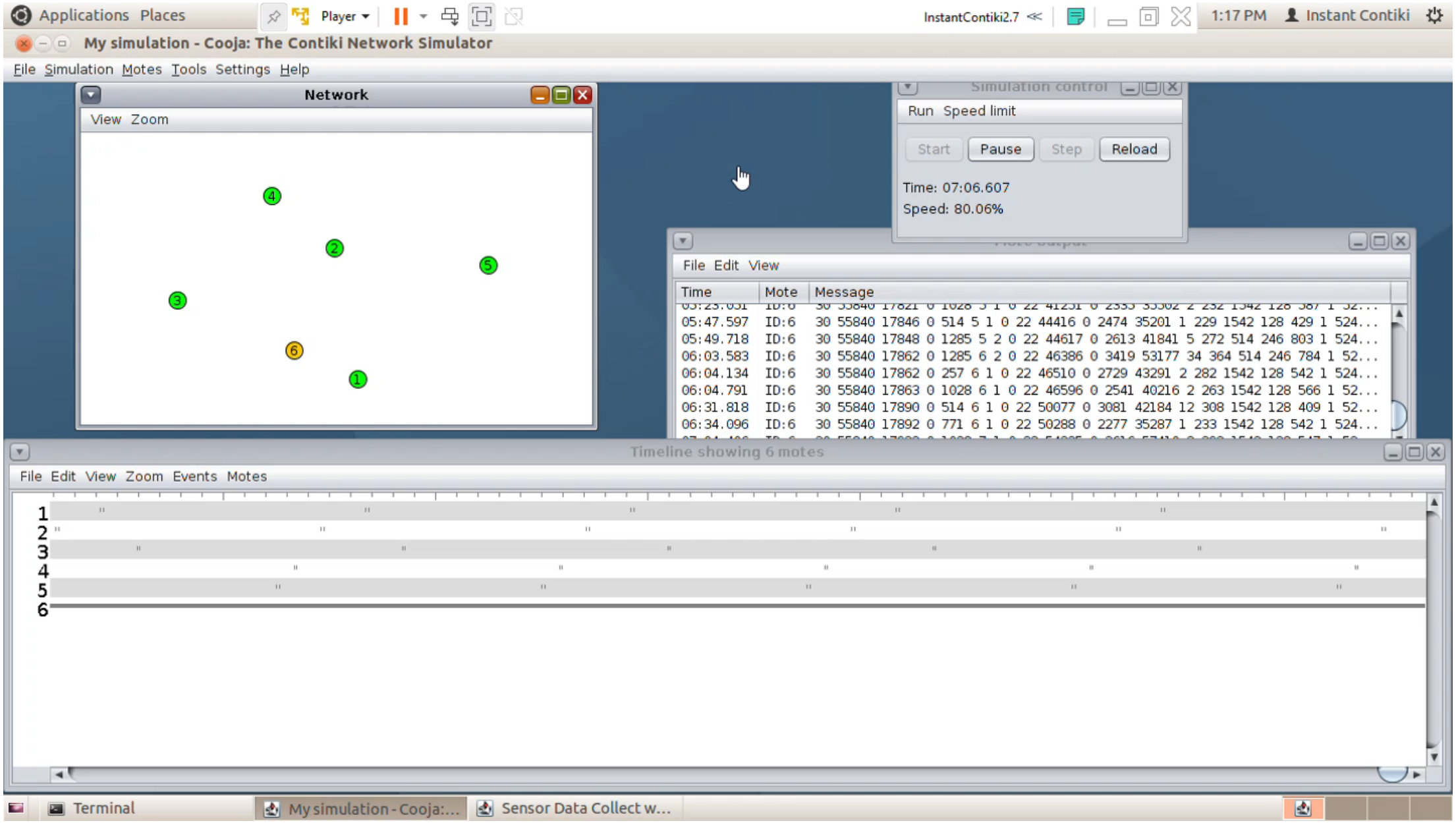


Algorithms	Accuracy	Model Training Time
Logistic Regression	41%	0.32
Decision Trees	74.5%	0.062
Random Forest	98.80%	49 sec
XGBoost	98.76	32 sec
LightGBM	99.28%	26 sec
Hist Gradient Boosting	99.37%	213 sec
CNN	98.8%	1200.5 sec
RNN	96.03%	1519 sec
LSTM	98.54%	1795 sec
GRU	98.80%	2701.Sec
Deep NN	98.74%	601.88sec
CNN + LSTM	98.70%	1582.07sec
Stacked ML	98%	2400sec

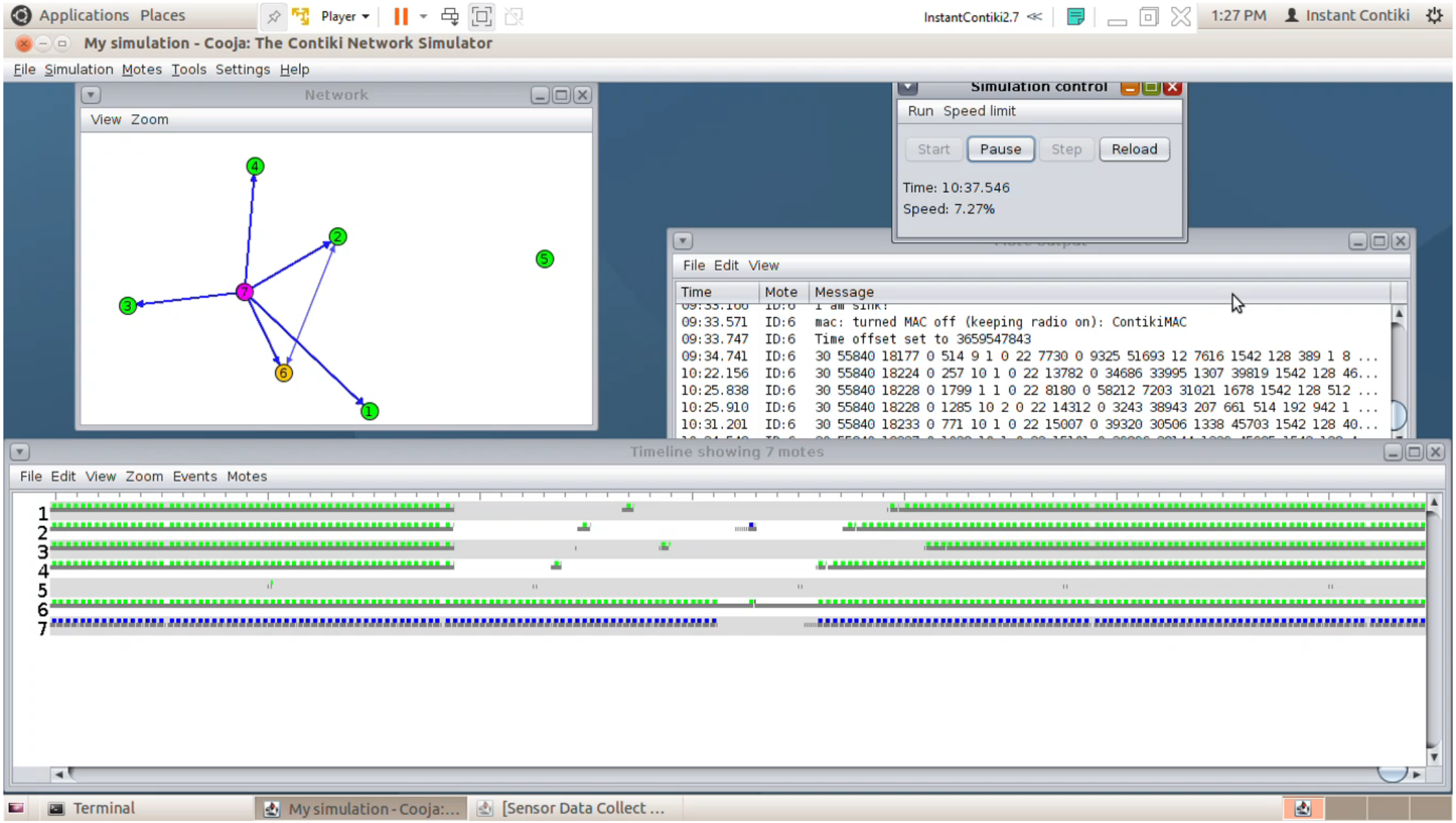
Showing one DoS Attack (UDP Flooding)

1. 5 Sender Nodes (Represented by Green Nodes)
2. 1 Base Station (Represented by Orange Node)
3. 1 Malicious Node with UDP Flooding packets (Represented by Violet Node)
4. Following a Star Topology
5. Used VMware workstation Player
6. Contiki Operating System
7. Cooja Simulator

Simulation without Attack



Simulation with Attack



Challenges

1. As there was no experimental setup for the traces collected from different sensors and services, we cannot deploy any kind of attacks.
2. In future, different classes of attacks can be deployed from laptop, which consists of all the data packets captured while communicating with different devices.
3. To make a trustworthy ML or DL models, it is required to make models robust and smart in any kind of situations.
4. Presently, the various ML and DL techniques used can’t guarantee their success in some of the critical conditions like in Healthcare, where if a cyber-attack occurs then it can also leads to death of a patient.

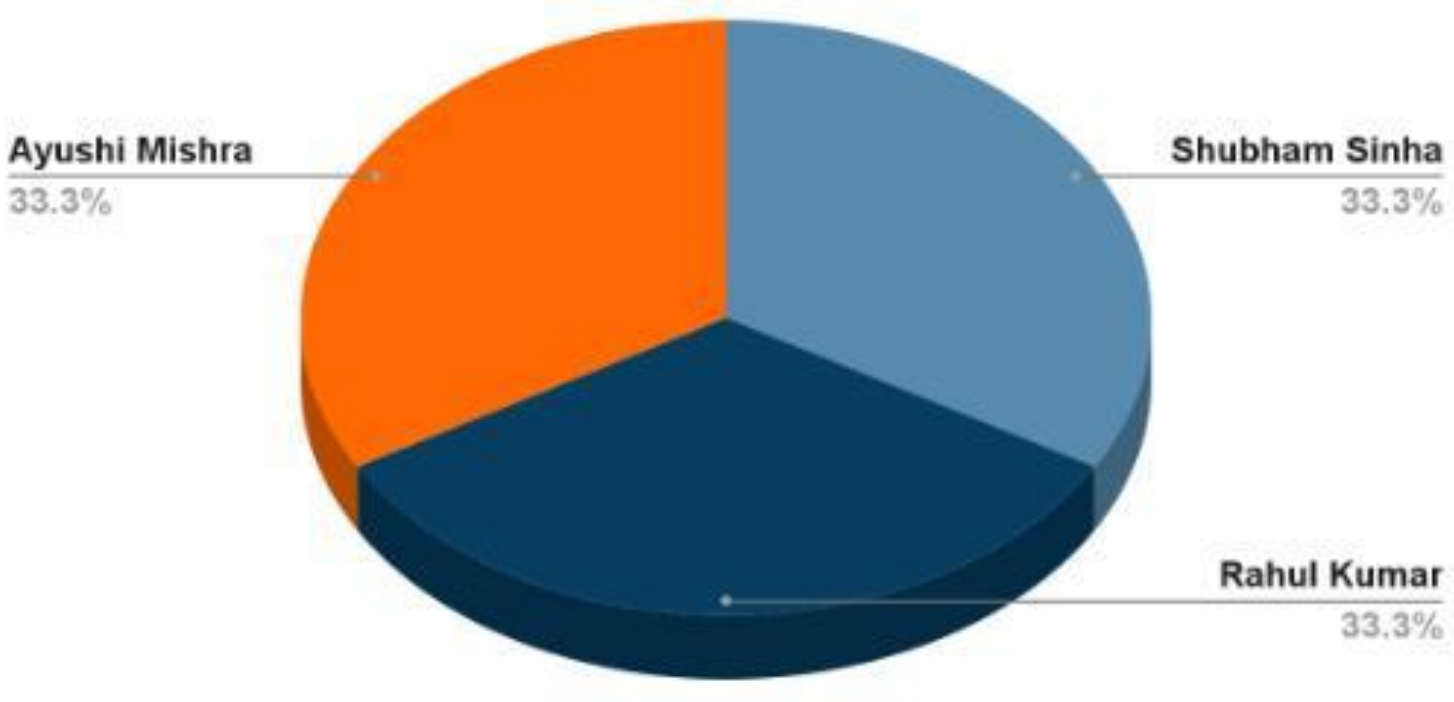
Future Work

1. After collection of data, we can deploy the attacks.

2. A notification can be send to the smartphone if a cyber-attack occurs.

3. We can use Digital Forensics to investigate the data and collect evidence. This will help in gathering information like which part of the system got affected and what kind of data, whether it was a file artifact or emails, audio and video files.

Individual Contribution



Thank You