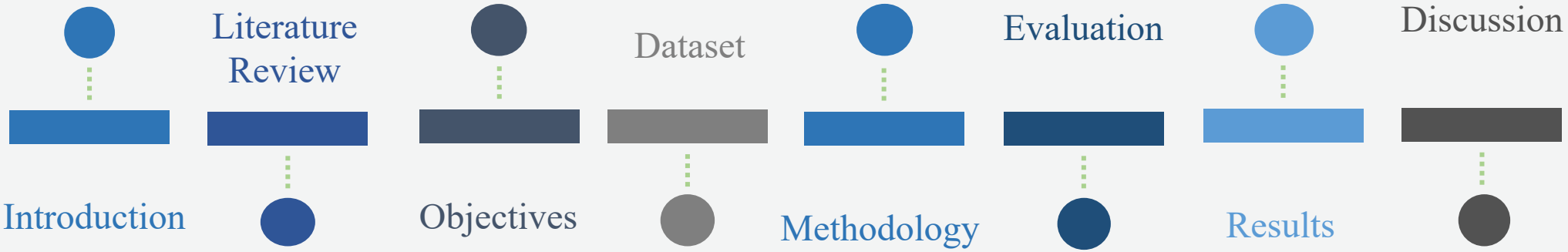
AN EFFICIENT ANDROID MALWARE PREDICTION USING ENSEMBLE MACHINE LEARNING ALGORITHMS

Neamat Al Sarah[1]
Fahmida Yasmin Rifat[1]
Md. Shohrab Hossain[2]
Husnu S. Narman[3]

[1]Department of Computer Science and Engineering, Military Institute of Science and Technology, Mirpur, Dhaka, Bangladesh.
[2]Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Bangladesh.
[3]Weisberg Division of Computer Science, Marshall University, Huntington, WV, USA.

Presentation Outline

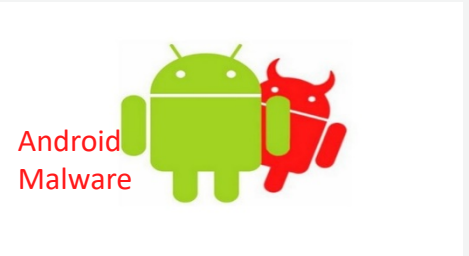


Bangladesh University of Engineering and Technology

MARSHALL UNIVERSITY

Introduction

- ✓ Lack of trustworthiness review methods, developers can upload their Android apps including repackaged apps, ransomware , or trojans to the market easily in even Google's Android market
- ✓ Which posed serious threats to the smart phone users, such as stealing user credentials, auto-dialing premium numbers, and sending SMS messages without user's concern



Objectives

- To automate the process of malware detection
- To detect malware within short time
- To achieve high accuracy with minimum number of false positives
- To be able to detect malware from all families
- To find out the best algorithm for predicting malware

Brief Literature Review

Static Analysis

Relies on features extracted without executing code

Static features such as manifest file components, API calls are used

Analysis for malware detection is done without running the app

Low resource consumption, fast detection and low real time requirements

Dataset

- ✓ We use DREBIN Dataset
- ✓ Contains 5560 applications from 179 different malware family
- ✓ Collected in the period of August 2010 to October 2012
- ✓ Available to us by the Mobile Sandbox project.

Eight Features of DREBIN dataset

Class	Name	Number
S1	Hardware Components	72
S2	Requested Permission	3,812
S3	App contents	218,952
S4	Filtered Intents	6379
S5	Restricted API Calls	733
S6	Used Permission	70
S7	Suspicious API Calls	315
S8	Network Address	210447

Bangladesh University of Engineering and Technology

MARSHALL UNIVERSITY

Methodology

Collecting apks

STATIC ANALYSIS

DREBIN

Dataset

Pre Processing

Training

Model

Prediction

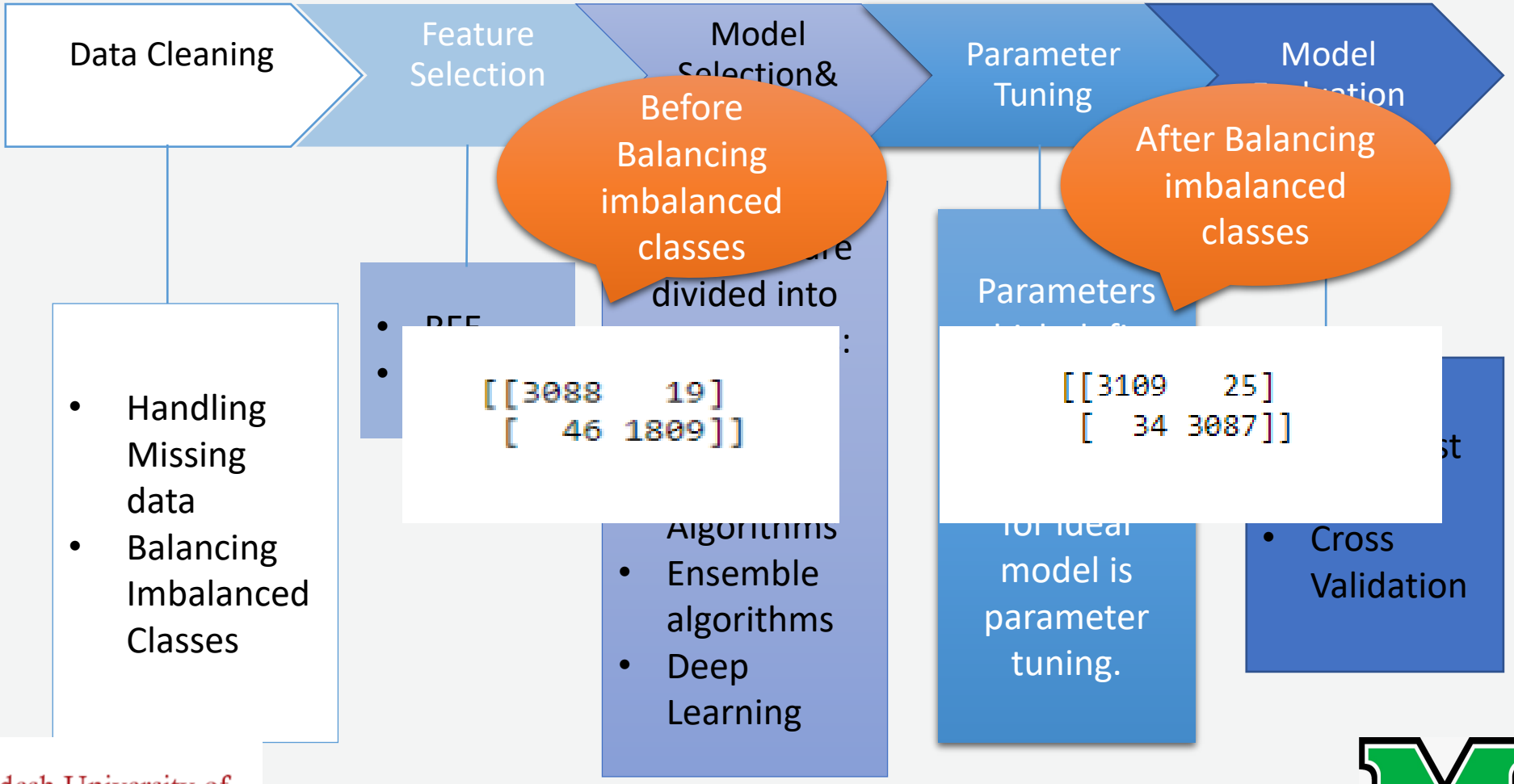
Malware

Benign

Bangladesh University of Engineering and Technology

MARSHALL UNIVERSITY

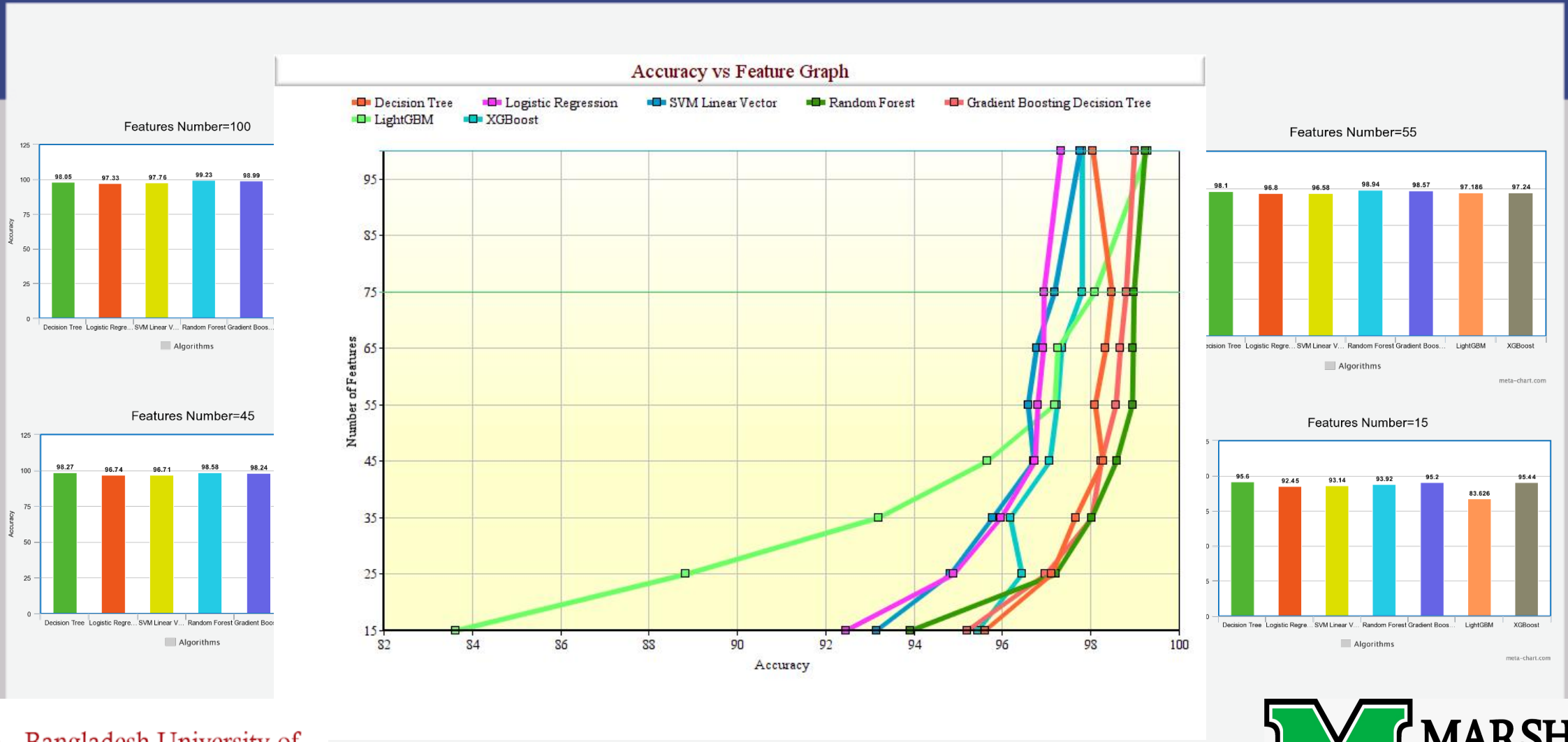
Our Approach: Key Features



Bangladesh University of Engineering and Technology

MARSHALL UNIVERSITY

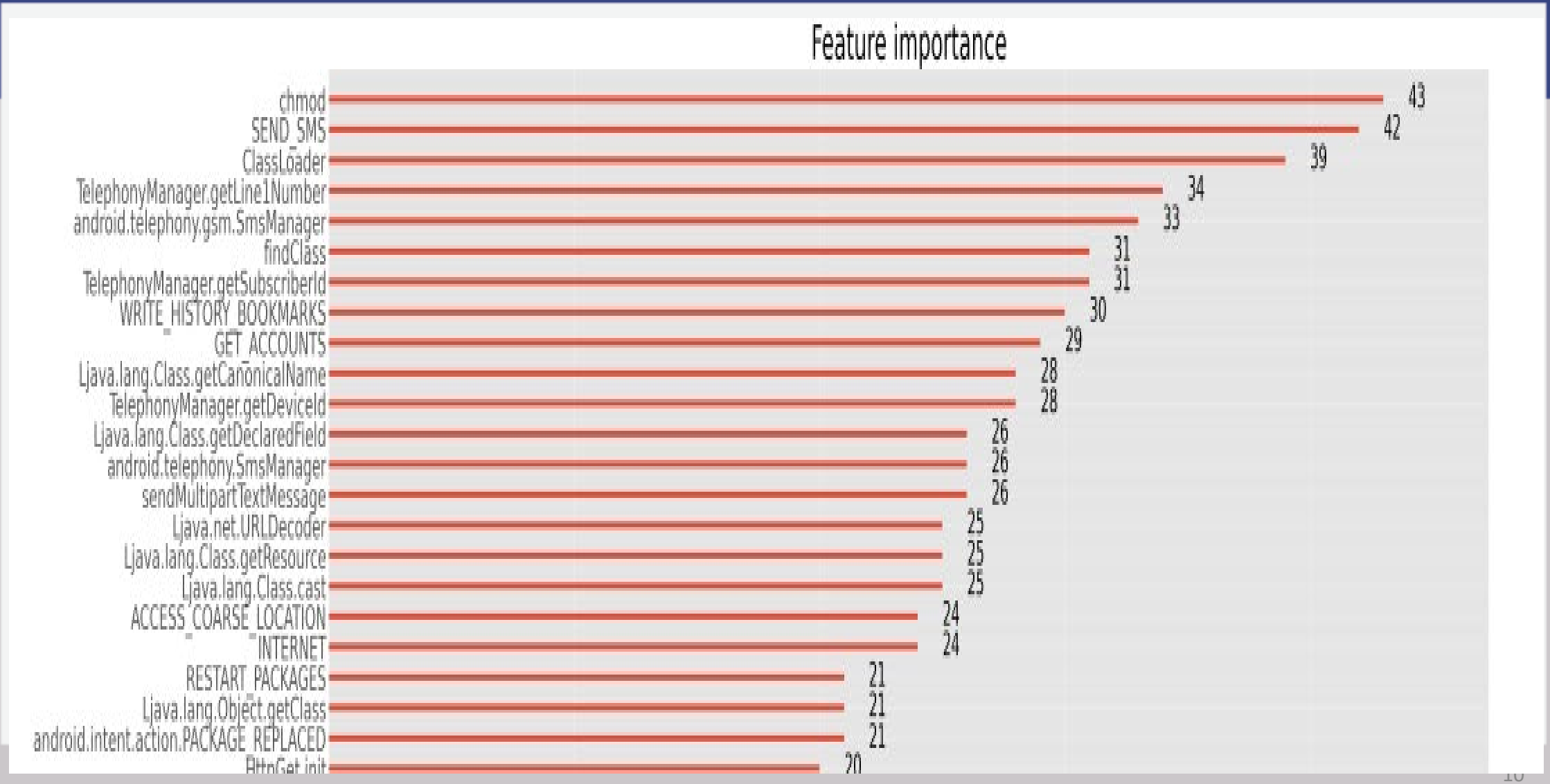
Feature Selection using RFE



Bangladesh University of Engineering and Technology

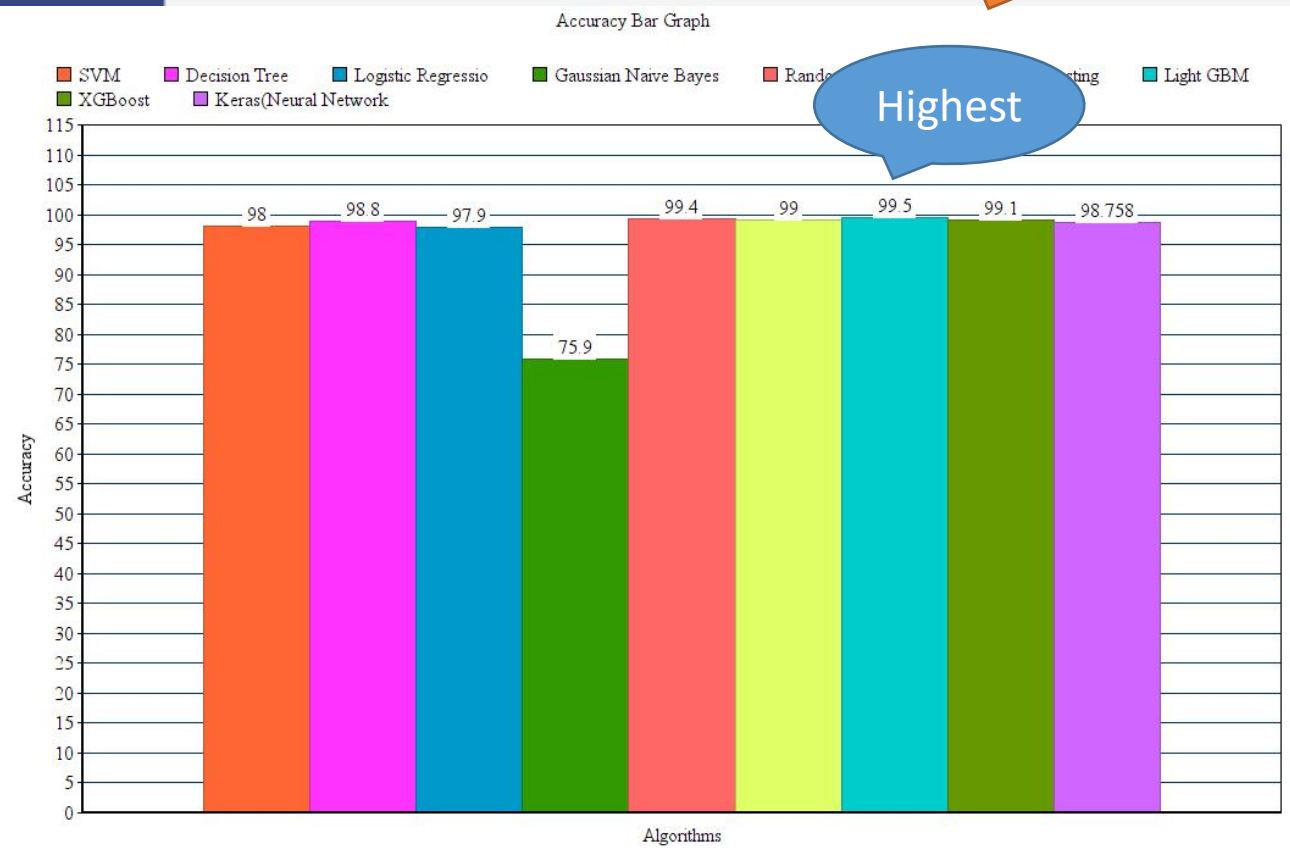
MARSHALL UNIVERSITY

Significant Features

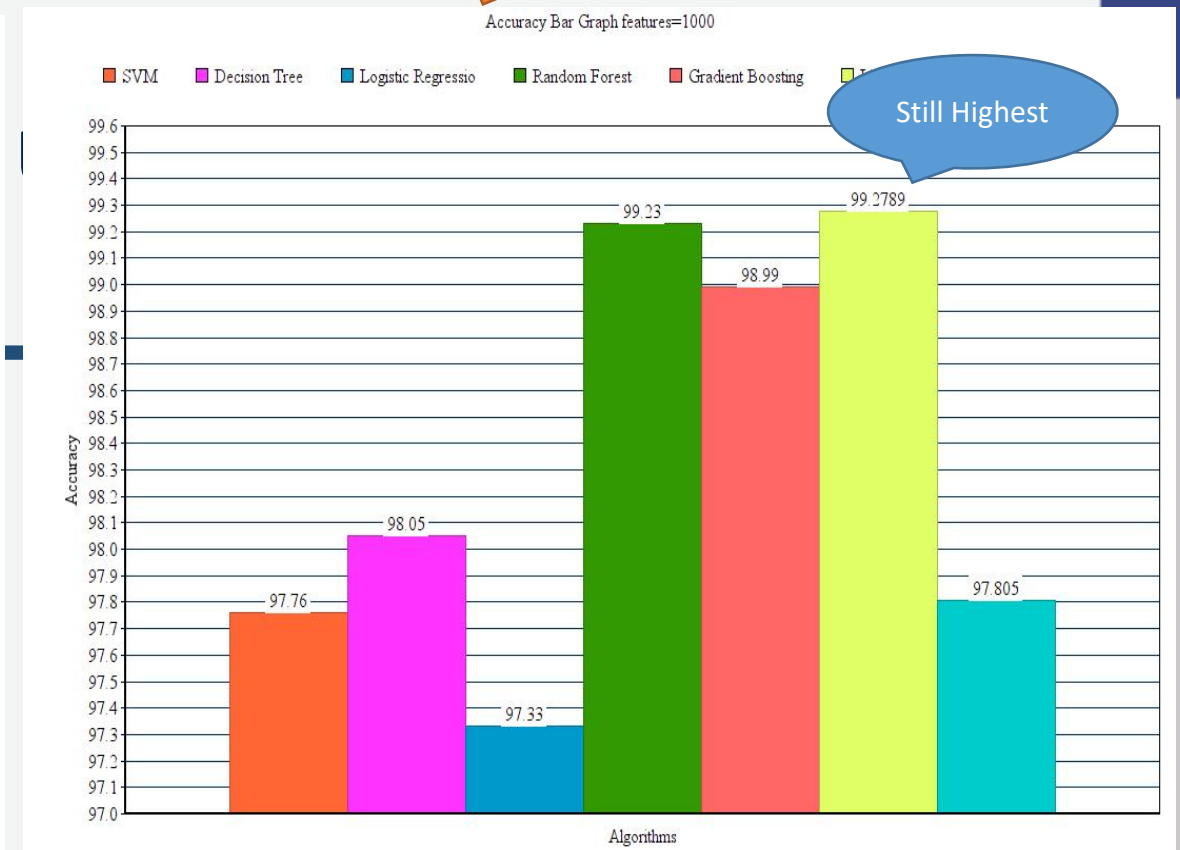


Model Training

Number of features=250



Number of features=100

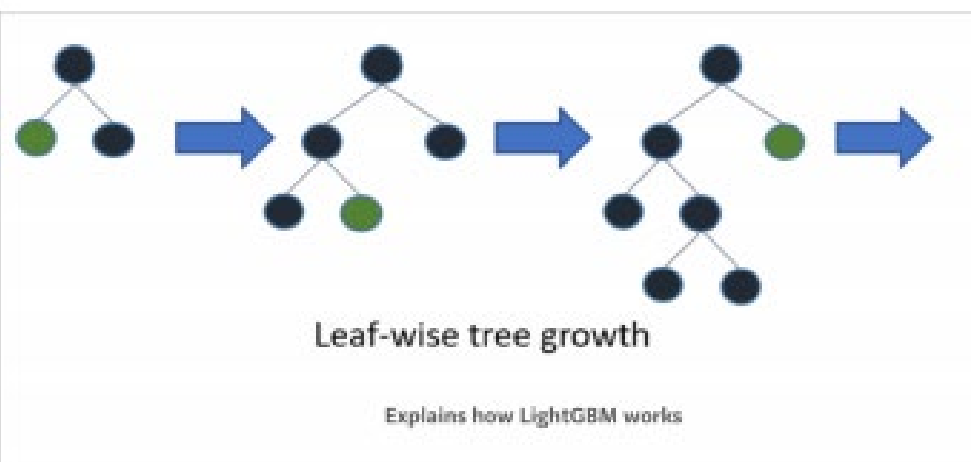


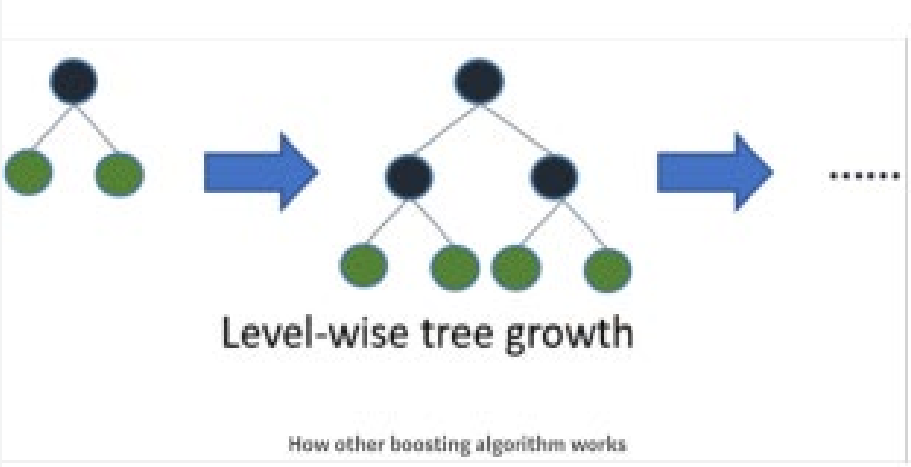
Bangladesh University of Engineering and Technology

MARSHALL UNIVERSITY

Why LightGBM?

- ✓ Gradient Boosting Ensemble Algorithm
- ✓ Fast, distributed, high-performance gradient boosting framework based on decision tree algorithm
- ✓ Grows tree leafwise while other algorithm grows level wise.
- ✓ Chooses the leaf with max delta loss to grow.
- ✓ When growing the same leaf, Leaf-wise algorithm can reduce more loss than a level-wise algorithm





Bangladesh University of Engineering and Technology

MARSHALL UNIVERSITY

Parameter Tuning

- ✓ Exploring a range of possibilities
- ✓ Parameters are crucial
- ✓ Finding an optimal combination of parameters that minimizes a predefined loss function to give better results

For Faster Speed

- bagging by setting bagging_fraction and bagging_freq
- feature sub-sampling by setting feature_fraction
- small max_bin
- save_binary to speed up data loading in future learning

Parameter Tuning

For Better Accuracy

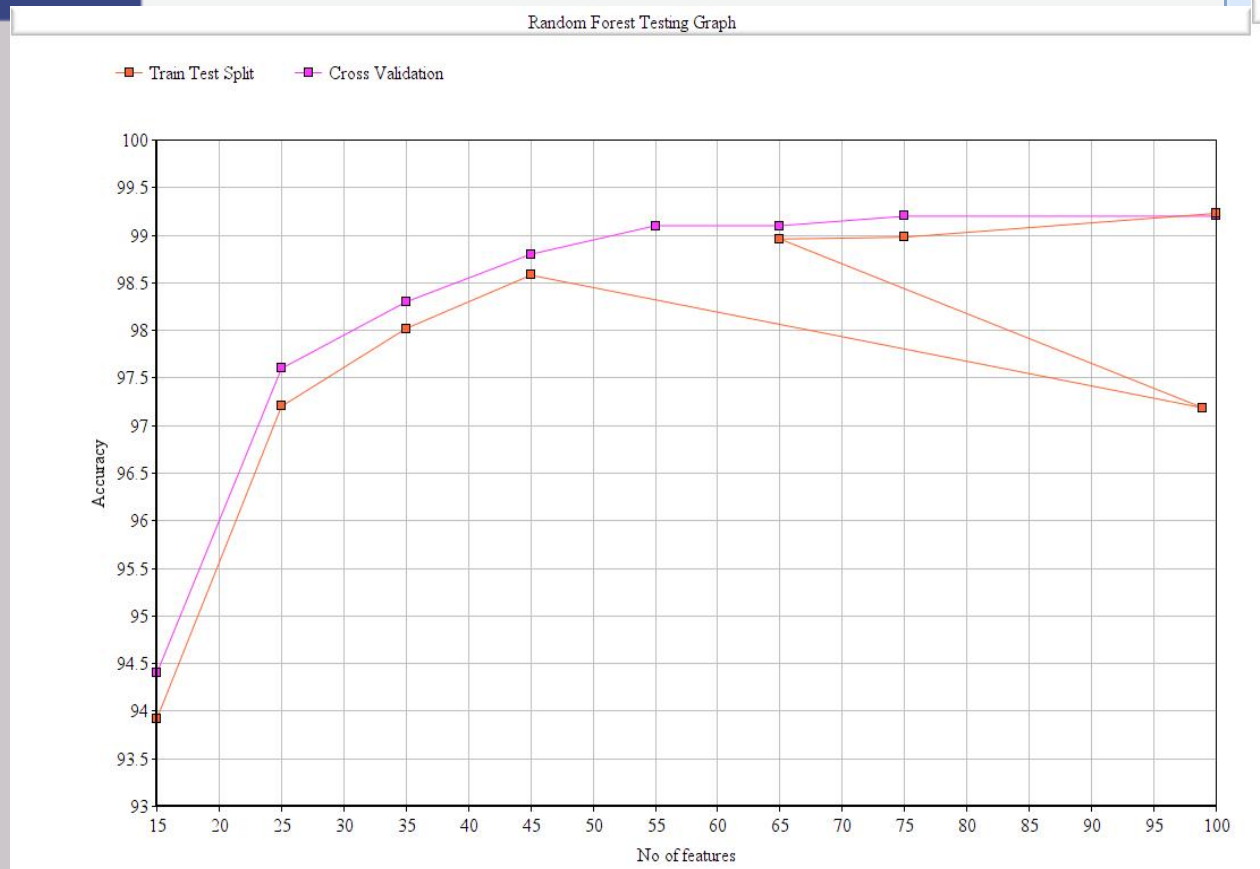
- ✓ large max_bin (may be slower)
- ✓ small learning_rate with large num_iterations
- ✓ large num_leaves (may cause over-fitting)
- ✓ bigger training data
- ✓ dart

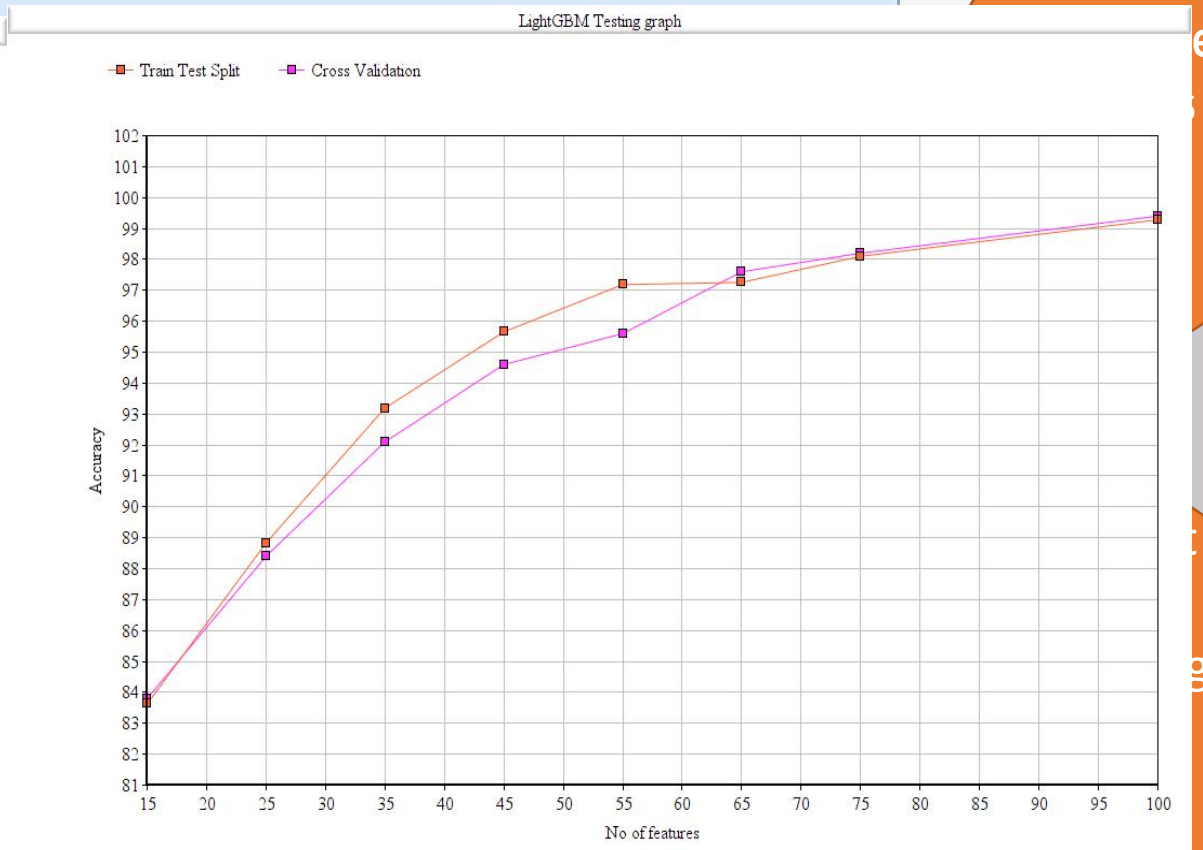
Deal with Over-fitting

- ✓ small max_bin
- ✓ small num_leaves
- ✓ min_data_in_leaf and min_sum_hessian_in_leaf
- ✓ bagging by set bagging_fraction and bagging_freq
- ✓ feature sub-sampling by set feature_fraction
- ✓ bigger training data
- ✓ lambda_l1, lambda_l2 and min_gain_to_split for regularization
- ✓ max_depth to avoid growing deep tree

Testing

Full dataset



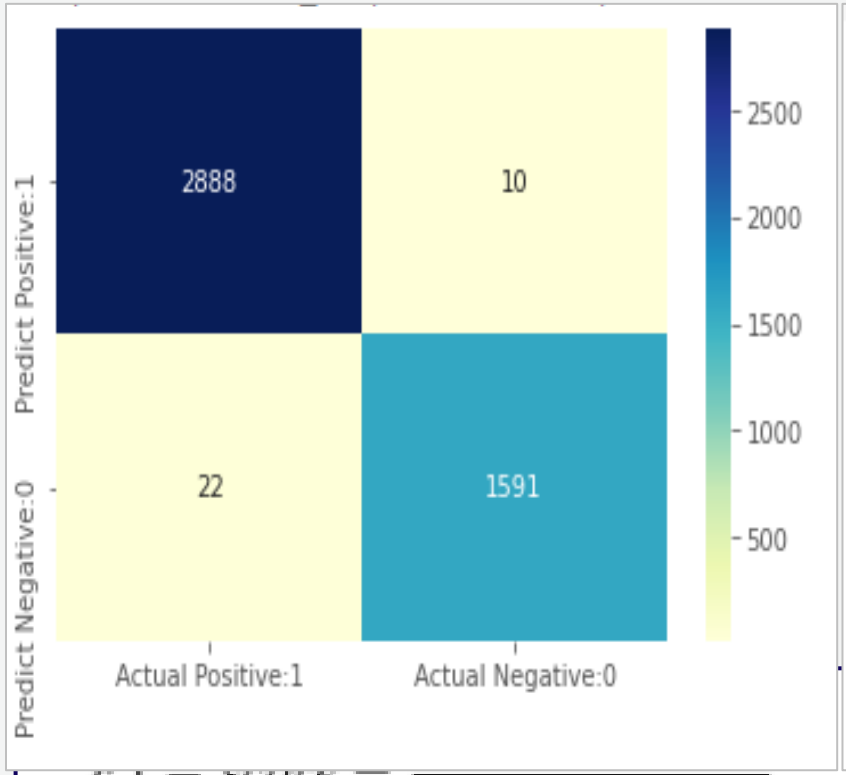


Training data

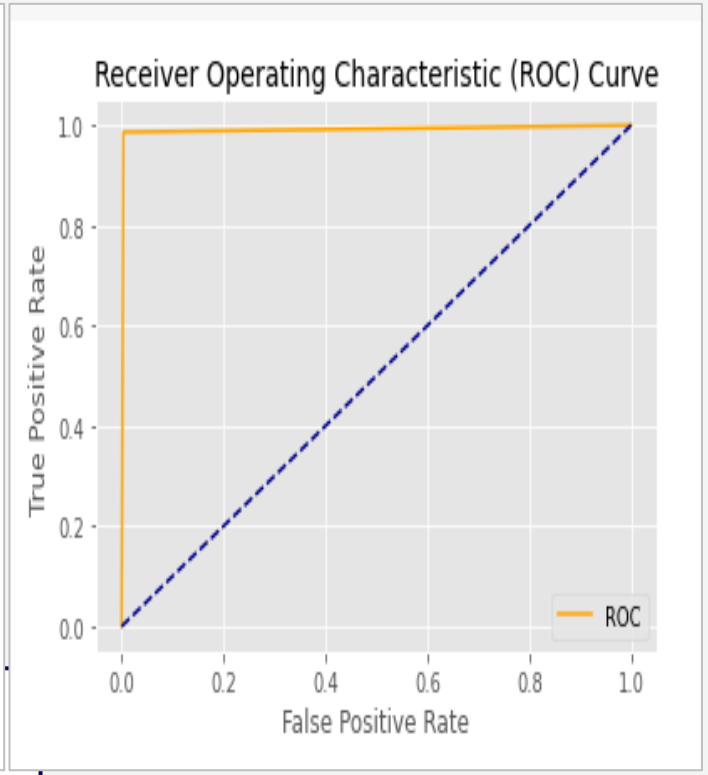
Test data

Evaluation Criteria & Results

Criteria Based Results



Confusion Matrix



ROC

1. Accuracy:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

0.9937

2. Recall:

$$Recall = \frac{TP}{TP + FN}$$

0.9924

0.991

3. F1 Score

$$F1 - score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precisio}$$

4. No. of false positive slices = False Detection

5. No. of false negative slices = Failed Detection

Bangladesh University of Engineering and Technology

MARSHALL UNIVERSITY

Thank You !

Bangladesh University of Engineering and Technology

MARSHALL UNIVERSITY