# Big Data - Assignment 3 (Group)

Magnus Bugge (300657712), Devon Grossett (300582913),
Joy Huixin Guan (300657179)

8 Jun 2024

## 1 Introduction

In this project, we trained a Decision Tree and a Logistic Regression model on the KDD Dataset [1] using Spark Machine Learning Libraries. The dataset contains 47,736 observations of simulated network intrusion on a military network environment, with 41 features to aid in prediction, and a label for "normal" or "anomaly" that we are trying to predict. The classes are well balanced in the data, with 24,576 labelled "anomaly" and 23,160 labelled "normal".

## 2 Program Pseudo-Code

```
Import Statements
if not enough arguments:
    exit program
initialize spark session
data := read csv(input argument)

set features and label variables
setup spark indexer over label variable
indexed data := fit, transform data with indexer
setup spark vector assembler over feature variables
assembled data := fit, transform indexed data with vector assembler

# Decision Tree Program
iterations := 10
initialize train, test, and time arrays
for i in {1, 2, ..., iterations}:
    random seed := i
    train, test := 80:20 split of data
    setup spark decision tree

    record start time
    fit decision tree to data
```

```
    record end time
    execution time := end time - start time
    append execution time to time array

    training predictions :=
        transform training data with decision tree
    test predictions :=
        transform test data with decision tree

    setup classification evaluator with accuracy metric
    training accuracy := evaluate training predictions
    append training accuracy to train array
    test accuracy := evaluate test predictions
    append test accuracy to test array

# Logistic Regression Program
iterations := 10
initialize train, test, and time arrays
for i in {1, 2, ..., iterations}:
    random seed := i
    train, test := 80:20 split of data
    setup spark decision tree

    record start time
    fit logistic regression to data
    record end time
    execution time := end time - start time
    append execution time to time array

    training predictions :=
        transform training data with decision tree
    test predictions :=
        transform test data with decision tree

    setup classification evaluator with accuracy metric
    training accuracy := evaluate training predictions
    append training accuracy to train array
    test accuracy := evaluate test predictions
    append test accuracy to test array

for {decision tree, logistic regression}:
    calculate min, max, mean, std-dev of train, test, and run time

store results in dataframe and write to disk
```

# 3 Readme

```
## Move Files to ECS System
This assumes you have a access to 'scp', which is available on Mac OS
or Linux. If you are running Windows, you can use WSL (Windows
Subsystem for Linux) or some other method to move the files onto the
network location

Move part1.py, kdd.data, and SetupSparkClasspath.sh to
barretts@ecs.vuw.ac.nz
$ scp part1.py <username>@barretts.ecs.vuw.ac.nz:
etc..

## Access VUW Hadoop cluster
ssh into barretts using your ecs account
$ ssh <username>@barretts.ecs.vuw.ac.nz

ssh into one of the Hadoop nodes
$ ssh co246a-5
(last number can be 1-8)

## Setup Hadoop and Spark
configure Hadoop and Spark
$ source SetupSparkClasspath.sh

create directory for input and output datasets
$ hadoop fs -mkdir /user/<username>/input /user/<username>/output

upload input data into hdfs
$ hadoop fs -put kdd.data /user/<username>/input/

## Run Spark Job
part1.py takes 2 inputs:
- path to input data
- path to output folder

$spark-submit --master yarn --deploy-mode cluster part1.py
/user/<username>/input/kdd.data /user/<username>/output

## Retrieve Results
move from hdfs to ecs local
$ hadoop fs -copyToLocal /user/<username>/output
$ hadoop fs -rm -r /user/<username>/output

move from ECS system to desired path local pc
$ scp -r <username>@barretts.ecs.vuw.ac.nz:~/output ~/path/to/local
```

# 4 Model Results and comparison

Table 1: Summary statistics for the training and test accuracy, and training time for a Decision Tree and Logistic Regression model on the KDD Dataset

| | Logistic Regression | | Decision Tree | |
|---|---|---|---|---|
| Measure | Training | Test | Training | Test |
| Min Accuracy | 0.887 | 0.885 | 0.947 | 0.943 |
| Max Accuracy | 0.891 | 0.891 | 0.956 | 0.955 |
| Mean Accuracy | 0.888 | 0.888 | 0.952 | 0.950 |
| Stdev Accuracy | 0.0012 | 0.0017 | 0.0029 | 0.0038 |
| Min Run-Time/s | 3.11 | - | 1.96 | - |
| Max Run-Time/s | 6.28 | - | 4.49 | - |
| Mean Run-Time/s | 4.67 | - | 2.28 | - |
| Stdev Run-Time/s | 0.92 | - | 0.74 | - |

Both models are being run with default settings. The results we obtained from these models on the KDD dataset are given in Table 1. As it can be seen in the "Mean Accuracy" row, both models' training and test accuracy are roughly equal, which means neither model is overfitting and the model generalises well to unseen data, which is desirable. The Decision Tree performs better on this dataset than Logistic Regression. The minimum test accuracy of the Decision Tree is 0.943, and the maximum test accuracy of the Logistic Regression is 0.891, so even in the worst case the Decision Tree model performs quite a bit better than the best case of the Logistic Regression. Both models have a very low standard deviation, which indicates there aren't any convergence issues, and that they have a very stable performance on this dataset.

The mean runtime of the Decision Tree model is half of the mean runtime for the Logistic Regression model, however both of them still only takes a couple of seconds, which is immaterial with the current size of the data. It is unknown how this performance scales with an increase in the size of the data, but if these proportions remained fairly consistent then the faster training time seen for the Decision Tree may also factor into it being a better choice of model when the training time becomes a determining factor.

# References

[1] S.J. Stolfo, Wei Fan, Wenke Lee, A. Prodromidis, and P.K. Chan. Cost-based modeling for fraud and intrusion detection: results from the jam project. In *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, volume 2, pages 130–144 vol.2, 2000.