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.txt, 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.txt /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.txt /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 Analysis

4.1 Results

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	-

4.2 Result Analysis

4.2.1 Accuracy

Both models are being run with default settings. The results we obtained from these models on the KDD dataset are given in Table 1. Exhibiting a clear edge, the Decision Tree model not only reaches higher accuracy levels—with a mean of 0.952 for training and 0.950 for testing—but also outpaces Logistic Regression, which maintains a mean accuracy of 0.888 across both datasets. While Decision Trees show a slightly higher variability in performance (a standard deviation of 0.0029 for training and 0.0038 for testing compared to Logistic Regression's 0.0012 and 0.0017, respectively), the model remains a strong contender due to its robustness across different trials.

4.2.2 Running Speed

When it comes to speed, Decision Trees take the lead, clocking in at an average of 2.28 seconds for training, which starkly contrasts with the 4.67 seconds Logistic Regression requires. This efficiency makes Decision Trees particularly appealing in environments where quick model deployment and rapid response are crucial.

4.2.3 Standard Deviations

For Logistic Regression, the standard deviations are remarkably low, 0.0012 for training and 0.0017 for testing, which underscores a high level of consistency across different model runs. This stability suggests that Logistic Regression, despite its lower mean accuracy, performs reliably under varying conditions within the

confines of its model structure. On the other hand, Decision Trees exhibit slightly higher variability, with standard deviations of 0.0029 for training and 0.0038 for testing. Although standard deviations of Decision Tree are higher than those of Logistic Regression, they still indicate a reasonable consistency, especially given the higher complexity and flexibility of Decision Trees in handling diverse data types. The increased standard deviation in Decision Trees can be viewed as a trade-off for their enhanced accuracy and adaptability to the dataset's multifaceted nature.

4.2.4 Conclusion

Decision Trees gain their advantage from an innate capability to manage datasets with diverse data types seamlessly. The KDD dataset, rich with categorical features like `"protocol_type"` and `"service"`, proves challenging for Logistic Regression. Typically, Logistic Regression anticipates linear relationships and prefers numerical inputs, demanding substantial preprocessing—like one-hot encoding—to handle categorical variables effectively. These necessary transformations not only bulk up the computational load but also complicate the model structure, potentially diluting its performance.

In contrast, Decision Trees naturally dissect data based on feature types, eliminating the need for the cumbersome preprocessing that Logistic Regression relies on. Their adeptness at navigating the complex structure of the KDD dataset, combined with their high accuracy and operational efficiency, positions them as the more fitting model for such diverse datasets. The consistency in their performance metrics from training to testing further attests to their stability and reliability.

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.