CSCI 8360 Data Science Practicum
Department of Computer Science
University of Georgia

# Project 2: Malware Classification

DUE: Thursday, September 15 by 11:59:59pm

Out September 1, 2016

## 1 Overview

Congratulations, you've all done admirably with the standard faire of document classification. The scoreboard is impressive, with the leader upwards of 96% and everyone else barely 2.5 points behind. All without using any utility packages that run on Spark.

Your next task is similar, but different: you're performing classification yet again, and will probably end up using similar techniques to the ones you developed yourselves in P1. However, this time, you'll be using hexadecimal binaries as documents, and attempting to classify them into one of several possible malware families. **Your goal is to design a large-scale classifier in Apache Spark that maximizes its classification accuracy against a testing dataset.**

For this project, we are using the data from the Microsoft Malware Classification Challenge, which consists of nearly *half a terabyte* of uncompressed data. There are no fewer than 9 classes of malware, but unlike the documents from P1, each instance of malware has one, and only one, of the following family categories:

1. Ramnit
2. Lollipop
3. Kelihos_ver3
4. Vundo

5. Simda
6. Tracur
7. Kelihos_ver1
8. Obfuscator.ACY
9. Gatak

The format of the data is somewhat different from before. All the documents are in hexadecimal format, in their own files (one file per document); these files are located here:

`https://s3.amazonaws.com/eds-uga-csci8360/data/project2/binaries/<file>`

and the S3 path:

`s3://eds-uga-csci8360/data/project2/binaries/<file>`

Each file looks something as follows:

```
...
00401060 53 8F 48 00 A9 88 40 00 04 4E 00 00 F9 31 4F 00
00401070 1D 99 02 47 D5 4F 00 00 03 05 B5 42 CE 88 65 43
00401080 6F 3D 4D 00 77 73 CD 47 21 A5 F0 48 87 8E 4A 00
00401090 DF 47 00 00 8A E6 41 4D 73 D7 4A 00 0B 45 00 00
...
```

The first hexadecimal token of each line, looking something like `00401060`, is just the line pointer and can be safely ignored (as these will appear throughout many of the files, not unlike articles in natural language). The other hexadecimal pairs, however, are the code of the malware instance itself and should be used to predict the malware's family.

Each binary file is independently identified by its hash, such as `01SuzwMJEIXsK7A8dQbl`. The files are named with their hashes (and have "`.bytes`" file extensions, after the hash; so a full filename would look like `01SuzwMJEIXsK7A8dQbl.bytes`); in a different directory are text files that contain these hashes, one per line, to indicate which files (or documents) are part of which dataset. The dataset definitions are located here:

`https://s3.amazonaws.com/eds-uga-csci8360/data/project2/labels/<file>`

and the S3 path:

`s3://eds-uga-csci8360/data/project2/labels/<file>`

Specifically, here are the available files in the `labels` directory:

- `X_train_small.txt`, `y_train_small.txt`
- `X_test_small.txt`, `y_test_small.txt`
- `X_train.txt`, `y_train.txt`
- `X_test.txt`

Each `X*` contains a list of hashes, one per line. Each corresponding `y*` file is a list of integers, one per line, indicating the malware family to which the binary file with the corresponding hash belongs.

There are "small" and "large" versions of the data available. The "small" is meant to help in initial algorithm development, as these data are small enough to fit on one machine. Only when you are confident in your core classification strategy should you move to the "large" dataset.

You'll notice there is no `y_test.txt` file. That file is being held on AutoLab, and will be used to evaluate your trained model and score your submission.

## 2 Guidelines

**Everyone must use Spark.** What language API you use is entirely up to you and your team. While you are still not allowed to use any packages that build *on top of* Spark, you are welcome to any and all Spark submodules: for example, MLlib and all its associated utilities.

**Create a GitHub repository for you and your team to work on the project.** You are more than welcome to make it a private repository, but the development must take place on GitHub, as I will be using it and the activity you record on it as part of the grading process. I would highly recommend using the repository naming scheme of `team-name-project2` to help differentiate from other teams and future projects. Make sure your repository includes 1) a README giving an overview of your program, how to install and run it, and your design rationale, 2) a MEMBERS file containing the names of you and your teammates, and 3) your code.

**Exercise good software design principles.** This means: adopt a consistent coding style, document your code, use the GitHub ticketing system to identify milestones and flag bugs, and provide informative and frequent `git commit` comments.

**Use flintrock or EMR.** You have two options here. You can either install flintrock for setting up and tearing down EC2 clusters for Spark, or you can use Amazon EMR (Elastic MapReduce) for spinning up ephemeral clusters in the AWS web dashboard.

**Shut down your EC2 clusters when you are finished testing.** AWS EC2 instances incur costs for every hour they are turned on, even if they are not actively running any jobs. I will not hesitate to revoke AWS credentials that are racking up disproportionate expenses. Here's the EC2 dashboard you can use to double-check if you have any resources allocated: `https://quinngroup.signin.aws.amazon.com/console`

## 3 Submissions

All submissions will go to **AutoLab**, our brand-new autograder and leaderboard system.

You can access AutoLab at `https://autolab.cs.uga.edu`. Your team will first need to create an account on AutoLab before you can be added to the `csci8360-fa16` course and submit assignments.

You can make submissions on behalf of your team to the **Project 2** assignment that is open. When you do, you will submit one file: **a text file containing the predictions of your trained classifier on the `X_test.txt` dataset, one prediction per line for each document**. For example, if you had six documents in the test set to classify, your output might look like:

```
1
7
7
3
5
3
```

Your classification accuracy will be tabulated and compared against the true labels (hidden on the server) and should appear on the leaderboard.

If you think you can do better than what shows up–particularly to beat out everyone else–make another submission! There's no penalty for additional submissions, but keep in mind that swamping the server at the last minute may result in your submission being missed; AutoLab is programmed to close submissions *promptly* at 11:59pm on Sept 15, so give yourself plenty of time!

## 4 Grading

Given that this is a practicum, and that you're all talented graduate students, the grading for the projects is a little different, operating on a sliding scale with some stationary points that require extra effort to move beyond.

As last time, everyone's starting grade is a **B**. Since you have all of MLlib at your disposal, you no longer need to code a classifier from scratch. Thus, it is a little trickier to define what constitutes the extra effort required for an **A**; the emphasis in this case will be more on the software engineering and *justification of methods used.* In your README, you should clearly explain 1) your classification rationale and design, 2) why it's appropriate for the problem, and 3) the advantages of your design over, say, the Naive Bayes method of P1.

You must demonstrate the extra customizations you made to your code and how those tweaks resulted in an improved score on the final testing set (getting the top score, for instance, would certainly qualify). Furthermore, you must show exemplary software engineering techniques: your code style should be clean, logical, and well-documented, and you should provide a clear README with instructions to new users on installing and deploying your document classifier. Finally, your `git commit` comments and GitHub tickets should provide a "paper trail" of documentation to show the route your team took in developing the software.

This combination of **algorithmic novelty** and **engineering best-practices** will earn you and your team that well-deserved **A**.

## 5 Reminders

- Your grade depends on two factors: the accuracy of the output you submit to AutoLab, and the quality of your code on GitHub. Don't neglect one!
- If you run into problems, work with your teammates. If you still run into problems, query the Slack channel. I'll be regularly checking Slack and interjecting where I can.
- This is the first time 8360 has been offered, and the first time AutoLab has been used, so there are bound to be hiccups. Please be patient! Thanks for your understanding.