Monash University

FIT5202 - Data processing for Big Data

Assignment 2: Detecting Linux system hacking activities

Due: Sunday, Oct 4, 2020, 11:55 PM (Local Campus Time)
Worth: 10% of the final marks

Background

StopHacking is a start-up incubated in Monash University to develop cloud service to detect and stop computer hackers. Although they have some rule-based service to identify certain hacks, they would like to add machine learning models which can integrate with their Spark cluster to process large amounts of data and detect any potential hacks. They hired us as *the Analytics Engineer* to investigate the open data from the Cyber Range Labs of UNSW Canberra and build models based on the data to identify abnormal system behaviour. In addition, they want us to help them integrate the machine learning models into the streaming platform using Apache Kafka and Apache Spark Streaming to detect any real-time threats, in order to stop the hacking. In this part A of the assignment, we would only need to process the static data and train machine learning models based on them.

What you are provided with

- Four data files:
  - Linux_memory_1.csv
  - Linux_memory_2.csv
  - Linux_process_1.csv
  - Linux_process_2.csv
- These files are available in Moodle in the Assessment section in the Assignment Data folder.
- A Metadata file is included which contains the information about the dataset.

Information on Dataset

Four data files recorded from Linux systems are provided, which captures information relating to the memory activity, and process activity. They are a subset of the Internet of Things dataset collected by Dr. Nour Moustafa, from IoT devices, Linux systems, Windows systems, and network devices. For more detailed information on the dataset, please refer to the Metadata file included in the assignment dataset or from the website below https://ieee-dataport.org/documents/toniot-datasets.

In this assignment, the memory activity and the process activity data are provided separately, without explicit linkage or computer ID to link up memory and process. For each data, there is a binary label and a multi-class label, one for indicating whether the activity is an attack or not, and the other for indicating which kind of attack the activity is under.

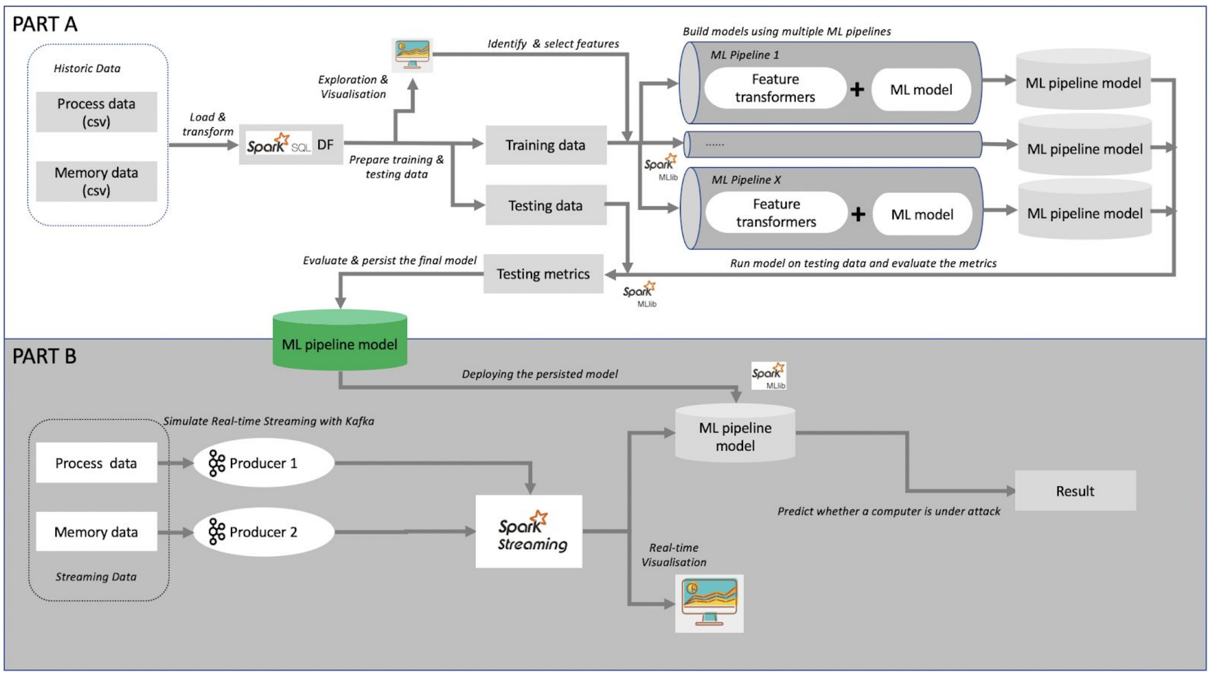
What you need to achieve

The StopHacking company requires us to build separate models for the memory activity and the process activity. Each activity needs a model for a binary classification predicting whether it is an attack or not, as shown in use case 1 & 2 below. To build the binary classification model, use the column “attack” as your label in the model (label 0 as non-attack event, and label 1 as attack event). You can select any columns as features from each activity data, except “attack” or “type”.

1	Process activity - attack or not	Binary classification
2	Memory activity - attack or not	Binary classification

Architecture

The overall architecture of the assignment setup is represented by the following figure. **Part A** of the assignment consists of preparing the data, performing data exploration and extracting features, building and persisting the machine learning models.



In both parts, for the data pre-processing, the machine learning processes , you are required to implement the solutions using PySpark SQL / MLlib / ML packages. For the data visualisations, please use Matplotlib packages to prepare the plots, and excessive usage of

Pandas for data processing is discouraged. Please follow the steps to document the processes and write the codes in Jupyter Notebook.

Getting Started

- Download the datasets from moodle.
- Create an ***Assignment-2A.ipynb*** file in Jupyter Notebook to write your solution for process data.

You will be using Python 3+ and PySpark 3.0 for this assignment.

1. Data preparation and exploration (35%)

1.1 Creating Spark Session (4%)

1. Create a SparkConf object for using as many local cores as possible, for a proper application name, and for changing the max partition byte configuration[1] to enable a minimum of 2 partitions[2] when reading each file in Spark SQL (so each dataframe should have at least 4 partitions when reading from the given datafiles).
2. Then create a SparkSession using the SparkConf object.

1.2 Loading the data (16%)

1. Load each activity data into a Spark dataframe and cache the data. Then print out the row count of each dataframe.
   - In order to speed up the loading process, please specify the schema before reading the data into dataframes. You may find relevant schema info from the metadata file, however, note that some data may not fully comply with the schema. For those that do not comply with the schema, import them as StringType and further transform them in step 1.2.2.
2. For each column in each dataframe above,
   - Check the null data (if any) and print out the corresponding count in each column
   - Are these columns 'MINFLT', 'MAJFLT', 'VSTEXT', 'RSIZE', 'VGROW', 'RGROW' in memory data following the datatype from the metadata file? If not, please transform them into the proper formats

1.3 Exploring the data (15%)

1. Show the count of attack and non-attack in each activity based on the column "attack", then show the count of each kind of attack in process activity based on the column "type".
   - Do you see any class imbalance? Examine and describe what you observe

[1] More details of configuration can be found on https://spark.apache.org/docs/latest/configuration.html
[2] This is a mock scenario considering the data size and the VM setup.

2. For each numeric feature in each activity, show the basic statistics (including count, mean, stddev, min, max); for each non-numeric feature in each activity, display the top-10 values and the corresponding counts.
    - No need to show the labels at “attack” or “type” column
3. For each activity, present two plots[3] worthy of presenting to the StopHacking company, describe your plots and discuss the findings from the plots
    - Hint - 1: you can use the basic plots (e.g. histograms, line charts, scatter plots) for relationship between a column and the “attack” label (such as “ts” and “attack”, “PID” and “attack”); or more advanced plots like correlation plots for relationship between each column; 2: if your data is too large for the plotting, consider using sampling before plotting
    - 100 words max for each plot’s description and discussion

2. Feature extraction and ML training (55%)

2.1 Preparing the training data and testing data (4%)

1. Randomly split the dataset into 80% training data and 20% testing data for each use case
2. With the class imbalance observed from 1.3.1, for the binary classification use case 1 & 2, prepare rebalanced training data, with attack events and non-attack events being 1:2 ratio, while using 20%[4] attack events data from the training data from 2.1.1. Cache the rebalanced training data, and display the count of each event's data.
    - Hint - you can use undersampling to get the rebalanced training data

2.2 Preparing features, labels and models (16%)

1. Based on data exploration from 1.3.3, which features would you select? Discuss the reason for selecting them and how you plan to further transform them[5].
    - 400 words max for the discussion
    - Hint - things to consider include whether to scale the numeric data, whether to choose one-hot encoding or string-indexing for a specific model
2. Create Transformers / Estimators for transforming / assembling the features you selected above in 2.2.1
    - (Bonus Task 5%) Create a custom Transformer for the column “POLI” so that the types of policy, [“norm”, “btch”, “idle”, “fifo”, “rr”, “0”, “-”], can be mapped to the following numbers, [0, 1, 2, 3, 4, 5, 6].

[3] This is an open question, in which you would need to decide what plots to show.
You can combine multiple features into one plot, but the plot should be clear to be seen, and do not contain an overwhelming amount of information.
If you use subplots, each subplot would be considered as one plot, and the two-plot limit would allow only two subplots for each activity data.
[4] This is a mock scenario considering the VM setup. Normally you do not need to downsize as much. Given a better cluster setup, you can use all the training data to carry out different model tuning strategies.
[5] This is an open question, in which you would need to decide what features to choose and what transformation(s) would be required for each feature. Include reference when you use arguments from third parties.

        i. Hint - you can create a custom Transform class inheriting from the PySpark ML Transformer, HasInputCol, HasOutputCol, DefaultParamsReadable, DefaultParamsWritable class, so that it can be included in the ML Pipeline in the next step. The class should allow users to specify the inputCol, outputCol, originalValues, and newValues parameters when initiating the object.

3. Prepare Estimators for Decision Tree and Gradient Boosted Tree model for each use case and include them into ML Pipelines for use case 1, 2
    - A maximum of two pipelines can be created for each use case

2.3 Training and evaluating models (35%)

1. For each use case, use the corresponding ML Pipeline from previous step to train the models on the rebalanced training data from 2.1.2
    - Hint - each model training might take from 1min to 40min, depending on the complexity of the pipeline model, the amount of training data and the VM computing power
2. For each use case, test the models on the testing data from 2.1.1 and display the count of each combination of attack label and prediction label in formats as below.

```
+------+----------+------+
|attack|prediction| count|
+------+----------+------+
|   1.0|       1.0|      |
|   0.0|       1.0|      |
|   1.0|       0.0|      |
|   0.0|       0.0|      |
+------+----------+------+
```

3. Compute the AUC, accuracy, recall and precision for the attack label from each model testing result using pyspark MLlib / ML APIs. Discuss which metric is more proper for measuring the model performance on identifying attacks.
4. Display the top-5 most important features in each model. Discuss which pipeline model is better, and whether the feature "ts" should be included in the model[6]. And visualise the ROC curve for the better model you selected for each use case.
    - 500 words max for the discussion
5. Using the pipeline model you selected in the previous step, re-train the pipeline model using a bigger set of rebalanced training data, with attack events and non-attack events being 1:2 ratio, while using all attack events data from the full data for both use cases. Then persist the better models for each use case.
    - The models would be deployed in Part B of the assignment 2.

3. Knowledge sharing (10%)

In addition to building the machine learning models, the IT manager from StopHacking would like to learn more about the internals of Spark ML, and plan to replace existing scikit learn

[6] This is an open question, in which you would need to present your argument and discussion. Include reference when you use arguments from third parties.

clustering logic by Spark KMeans clustering to cater large amounts of data. You are expected to combine the theory from the lecture and the observation from Spark UI to explain what happens when training the KMeans clustering model.

3.1 How many jobs are observed when training the KMeans clustering model following the code below? Provide a screenshot from Spark UI for running a simple KMeans model training. (0.5%)

- For example, run the following code,

  ```
  iris_df = spark.createDataFrame([
      (4.7, 3.2, 1.3, 0.2),
      (4.9, 3.1, 1.5, 0.1),
      (5.4, 3.9, 1.3, 0.4),
      (5.0, 3.4, 1.6, 0.4),
      (5.1, 3.8, 1.6, 0.2),
      (4.9, 2.4, 3.3, 1.0),
      (6.6, 2.9, 4.6, 1.3),
      (5.6, 3.0, 4.5, 1.5),
      (5.7, 2.6, 3.5, 1.0),
      (5.8, 2.6, 4.0, 1.2),
      (5.8, 2.8, 5.1, 2.4),
      (6.2, 2.8, 4.8, 1.8),
      (6.0, 3.0, 4.8, 1.8),
      (6.7, 3.1, 5.6, 2.4),
      (6.7, 3.0, 5.2, 2.3),
      (6.2, 3.4, 5.4, 2.3)],
      ['sepal_length', 'sepal_width', 'petal_length', 'petal_width'])

  assembler = VectorAssembler(
      inputCols=['sepal_length', 'sepal_width', 'petal_length', 'petal_width'],
      outputCol='features')
  kmeans = KMeans(k=3).fit(assembler.transform(iris_df))
  ```

3.2 Combining the KMeans algorithm from the lecture, explain what each job in Spark UI represents (9.5%)

- 300 words max for the discussion
- Hint - you can also refer to the Spark source code on github https://github.com/apache/spark/blob/master/mllib/src/main/scala/org/apache/spark/mllib/clustering/KMeans.scala

Assignment Marking

The marking of this assignment is based on quality of work that you have submitted rather than just quantity. The marking starts from zero and goes up based on the tasks you have successfully completed and it's quality for example how well the code submitted follows *programming standards, code documentation, presentation of the assignment, readability of the code, organisation of code and so on*. Please find the PEP 8 -- Style Guide for Python Code here for your reference.

Submission

You should submit your final version of the assignment solution online via Moodle; You must submit the following:

- A zip file of your Assignment 2A folder, named based on your authcate name (e.g. psan002). This should contain
    - **Assignment-2A.ipynb**

  This should be a ZIP file and *not any other kind of compressed folder (e.g. .rar, .7zip, .tar)*. Please do not include the data files in the ZIP file.
- The assignment submission should be uploaded and finalised by Sunday, Oct 4, 2020, 11:55 PM (Local Campus Time).
- Your assignment will be assessed based on the contents of the Assignment 2 folder you have submitted via Moodle. When marking your assignments, we will use the same ubuntu setup as provided to you in Week 01.

Other Information

Where to get help

You can ask questions about the assignment on the Assignments section in the Ed Forum accessible from the on the unit's Moodle Forum page. This is the preferred venue for assignment clarification-type questions. You should check this forum regularly, as the responses of the teaching staff are "official" and can constitute amendments or additions to the assignment specification. Also, you can visit the consultation sessions if the problem and the confusions are still not solved.

Plagiarism and collusion

Plagiarism and collusion are serious academic offences at Monash University. Students must not share their work with any other students. Students should consult the policy linked below for more information.

https://www.monash.edu/students/academic/policies/academic-integrity

See also the video linked on the Moodle page under the Assignment block.

Students involved in collusion or plagiarism will be subject to disciplinary penalties, which can include:

- The work not being assessed
- A zero grade for the unit
- Suspension from the University
- Exclusion from the University

Late submissions

Late Assignments or extensions will not be accepted unless you submit a special consideration form. ALL Special Consideration, including within the semester, is now to be submitted centrally. This means that students MUST submit an online Special Consideration form via Monash Connect. For more details please refer to the **Unit Information** section in Moodle.

There is a **5% penalty per day including weekends** for the late submission.