Assignment 1 Documentation

Samuel Price – pric0107

Random Forest for UNSW botnet dataset

NOTE: The FLO submission box currently doesn't allow .zip files to be uploaded, so this file is being uploaded by-itself to show I've done it on time and can't upload the program due to this.

Data Used:

Originally my data set is a network scan looking at different types of botnet attacks,

This data is available via UNSW at: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/bot_iot.php

A botnet is a type of cyber security attack which involves subtly using a user's computer to perform malicious activities, it involves two main hosts types

- a host computer (which is directing the attacks)
- zombie machine/s (network of zombie computers perform attacks)

these types of attacks are vast; however this dataset only contains traffic data for:

- DDoS
- DoS
- Malicious scans
- Keylogging and data exfiltration

The data is a network scan which logs all traffic coming in & out of a zombie computer – this isn't a real scenario but is meant as a simulation (performed in a private network environment) for training purposes.

The columns of this dataset are:

pkSeqID proto saddr sport daddr dport seq stddev N_IN_Conn_P_SrcIP

min state_number mean N_IN_Conn_P_DstIP drate srate max attack category subcategory

In my model I decide to drop the columns 'pkSeqID', 'attack' and 'subcategory' and I feel like I could drop more however I just let the model train with only dropping those.

The sample dataset for Iris flower classification was also used as this is more smaller to work with and has known results.

Model:

Random forest & Decision tree were chosen

I came across the botnet dataset first and decided to build a model around it -

There's actually a couple papers looking a specifically decision trees [1] or specifically random forests [2, 3] for network intrusion and attacks. Pretty much the same reasoning for looking at traffic

coming into the network is the same as looking at traffic going out (as in a zombie performing an attack)

The paper 'Botnet detection based on network behavior': 'Botnet detection' [1] looks specifically for packets pertaining to the IRC chat sessions (on port 6667) it also uses a J48 (java C4.5) decision tree. Differs slightly from my implementation from these.

The main paper that inspired this implementation was 'Support vector machine and random forest modeling for intrusion detection system' [3] which develops both a SVM and random forest implementation for the task of network intrusion detection, network intrusion is pretty much the reverse of botnet detection as explained before so I figured it could similarly be applied. This paper does give both a model for SVM and Random Forest with fairly similar results, SVM taking more time to train however – ultimately I decided to go with a random forest implementation as I simply wanted to develop a random forest model.

'Random forest modeling for network intrusion detection system' [2] is another paper that looks at using random forest for network intrusion detection, very similar to the paper above [3] with different explanation of how the model works. As with the paper above [3] this paper [2] looks at classifying types of attacks (i.e. DoS, U2R) which is what my model does – differences being these two papers looked at intrusion (coming into network) where my model looks at network going out being (being used as a zombie for attacks)

Code Documentation:

Novelties:

All major metrics/variables are located up the top of random_forest.py, you can change these in the code, however when you run the program, it will ask you if you want to change them for that run.

I've made it to be multi-threaded – really it's called multiprocessing in python. Quick explanation multi-threading in python doesn't really work, the python process gets locked to a single thread. Multiprocessing essentially creates new processes that'll run in parallel of each other, which is the typical multi-threading. This works well in linux but hasn't been tested on windows yet.
This by default is turned off, but when you run the program it will ask you if you want to turn it on, for both/either the random forest stage or predicting stage. Processes are based on the number of trees in the forest.

The only metric you have to manually change in the source file is data to load in, where to split and what columns to cut out.

For unsw_botnet dataset there's an attack label – which is just 1 or 0 based on whether it's an attack or not and a category label which says what type of attack the zombie pc is participating in. I classify by type of attack, which includes non-attacks/normal traffic.

To test accuracy of the program, it'll simply split up whatever csv file is loaded in, into a training and testing set. Accuracy is a value between 0 an 1, with differing metrics I set up, I get a result of anywhere between 0.86 which is 86% and 0.990 which is 99%. Best is around default.

The main dataset is too large to fit in the file submission size on FLO – therefore the test dataset included in the zip - unsw_10_test.csv is just a cut-down version of the original dataset used in testing - UNSW_10_BEST.csv available at: https://www.unsw.adfa.edu.au/unsw-canberra-

cyber/cybersecurity/ADFA-NB15-Datasets/bot_iot.php **Note:** that the data reads in this test file, simply uncomment out the line below to load in the original.

Files:

There are 5 files in the zip which are:

- **Iris.csv** – iris classification dataset
- **Unsw_10_test.csv** – unsw botnet test dataset
- **Dt_func.py** – Decision tree functions
- **Decision_tree.py** – Decision tree main file
- **Random_forest.py** – Random Forest main file & extension of dt_func.py

There are two separate models provided, a decision tree and random forest – I decided on this as I had already made the decision tree functions for the random forest, both are useable from their main file.

The decision tree model simply uses the Iris.csv dataset as it takes too long to classify unsw_botnet, while random forest uses unsw_10_test.csv

Where is the executable (.exe)? – This is a very last-minute note, I spent way too long trying to create a working executable and it wasn't working, so to run the file you need to have python installed, with the dependencies listed below. – if this paragraph is still in it means I ran out of time and would of just submitted without the .exe - **Run with the command** (while in the folder): python3 random_forest.py

Import Dependencies:

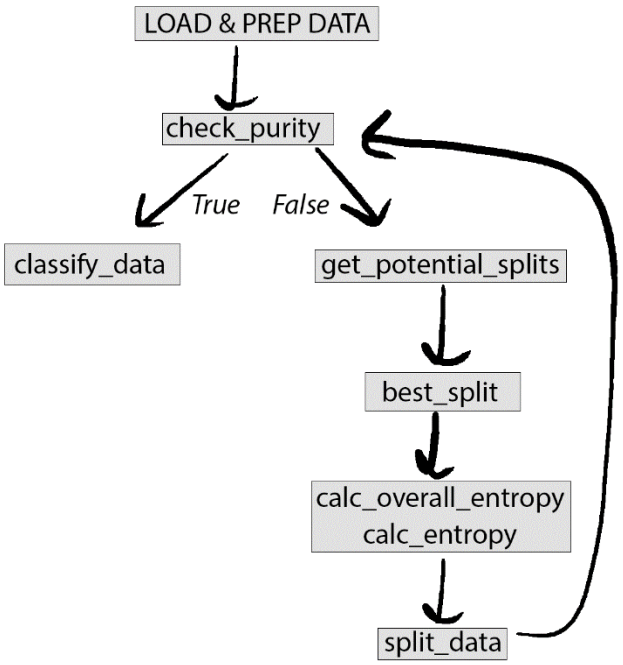
All import dependencies can be installed with pip (python package manager), though not every import needs to be installed as they come with python already.

The imports are:

- **Numpy** – numpy adds support for arrays/lists/matricies
- **Pandas** – used for dataframes and series
- **sklearn import preprocessing** – Sklearn does a lot of things, this code only uses it's preprocessing import, which is used for encoding data.
- **tqdm** – provides progress bars
- **pprint** – prints out trees more nicer
- **multiprocessing.dummy import Pool** – used for multiprocessing/threading
- **multiprocessing** – used for multiprocessing/threading
- **time** – used for timing how long it takes

Note: on my windows (worked fine on linux) I was having trouble installing **sklearn import preprocessing** this is only used for encoding the unsw_botnet dataset, if you're having this issue comment/uncomment to change it to load in iris.csv instead for testing.

The Decision Tree follows the diagram below:



LOAD & PREP DATA:

Name of the process of reading in data and preparing it – this stage is done in the Radom Forest file

1. **Read** - Data will be read in through a csv (unsw_10_test.csv or iris.csv),
2. **Drop** - specific columns will be dropped, in the case for unsw_train.csv the columns 'pkSeqID', 'attack' and 'subcategory' will be dropped.
3. **Rename** - Label column will be renamed to 'label', for unsw_train.csv this is the 'category' column.
4. **Encode** - Specific columns will be encoded – right now this is only needed for unsw_train.csv as for example IP Addresses
5. **Train/Test split** – the code will just read in one dataset and split it up into two separate data-frames for Training and Testing. This is just to make testing the code easy for me for testing.

Check_purity:

This function simply checks if a data-frame is pure – as in it only has a single label

- **True** – Move onto classifying the data-frame
- **False –** Split up data-frame

Get_potential_splits:

This function will return a list of all potential splits – achieves this by running through all columns and storing unique values into the list – used for best_split function

Best_split:

Uses the functions below (calc_overall_entropy) to calculate entropy of potential splits to determine where to split.

Calc_overall_entropy & calc_entropy:

The below functions for entropy are the standard ID3 Decision Tree entropy functions, they were obtained from: https://www.thelearningmachine.ai/tree-id3

Calc_entropy:

P(x) is the probability

$$E(S) = \sum_{x \in X} -P(x) \log_2 P(x)$$

Calc_overall_entropy:

$$E(S, A) = \sum_{x \in X} [P(x) * E(S)]$$

Split_data:

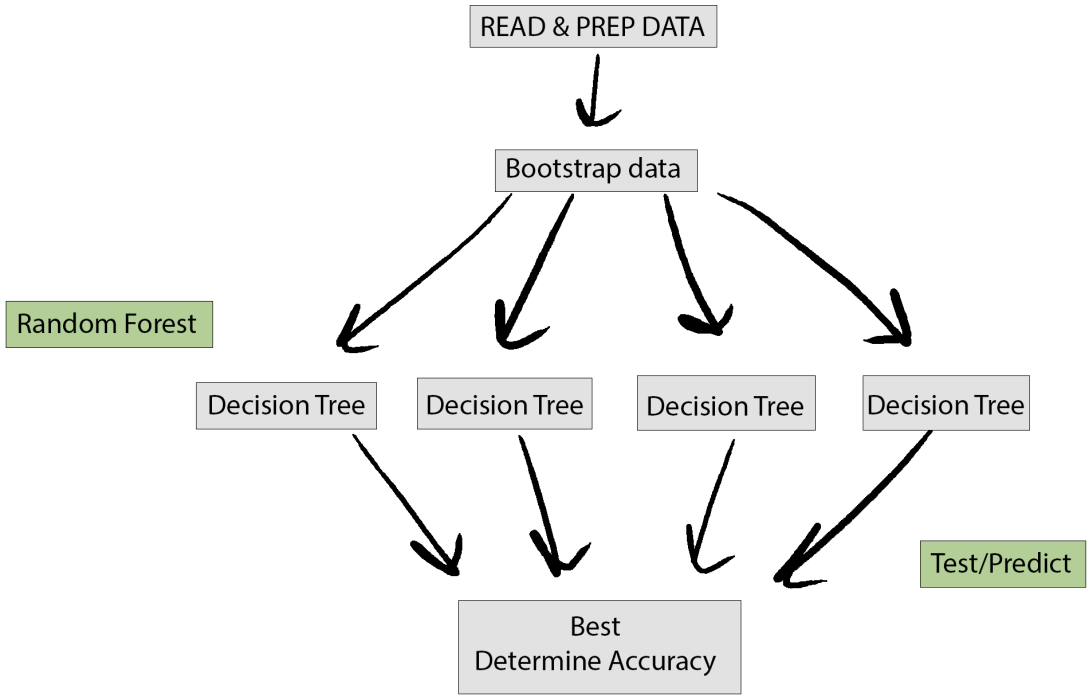
Just splits based on best_split results, returns data_below and data_above where split occurred.

And repeat – send the split dataframes to check_purity

Classify_data:

If check_purity is true the data that is pure will then be classified, this is the final stage of the main algorithm, after all has been classified the program will move onto testing/prediction.

The random forest algorithm looks like



Bootstrap:

Randomly selects data, write down into sample and return that

Random_forest:

This function uses bootstrap function above, creates trees with this sample and puts on a list. This has the option to be multiprocessed

Predict:

Uses classification function in Dt_funcs to predict a test series (from the train/test split) and the different trees – this has the option to be multiprocessed. The accuracy is determined based on these predictions

References:

1 Strayer, W.T., Lapsely, D., Walsh, R., and Livadas, C.: 'Botnet detection based on network behavior': 'Botnet detection' (Springer, 2008), pp. 1-24

2 Farnaaz, N., and Jabbar, M.J.P.C.S.: 'Random forest modeling for network intrusion detection system', 2016, 89, pp. 213-217

3 Hasan, M.A.M., Nasser, M., Pal, B., Ahmad, S.J.J.o.I.L.S., and Applications: 'Support vector machine and random forest modeling for intrusion detection system (IDS)', 2014, 6, (01), pp. 45