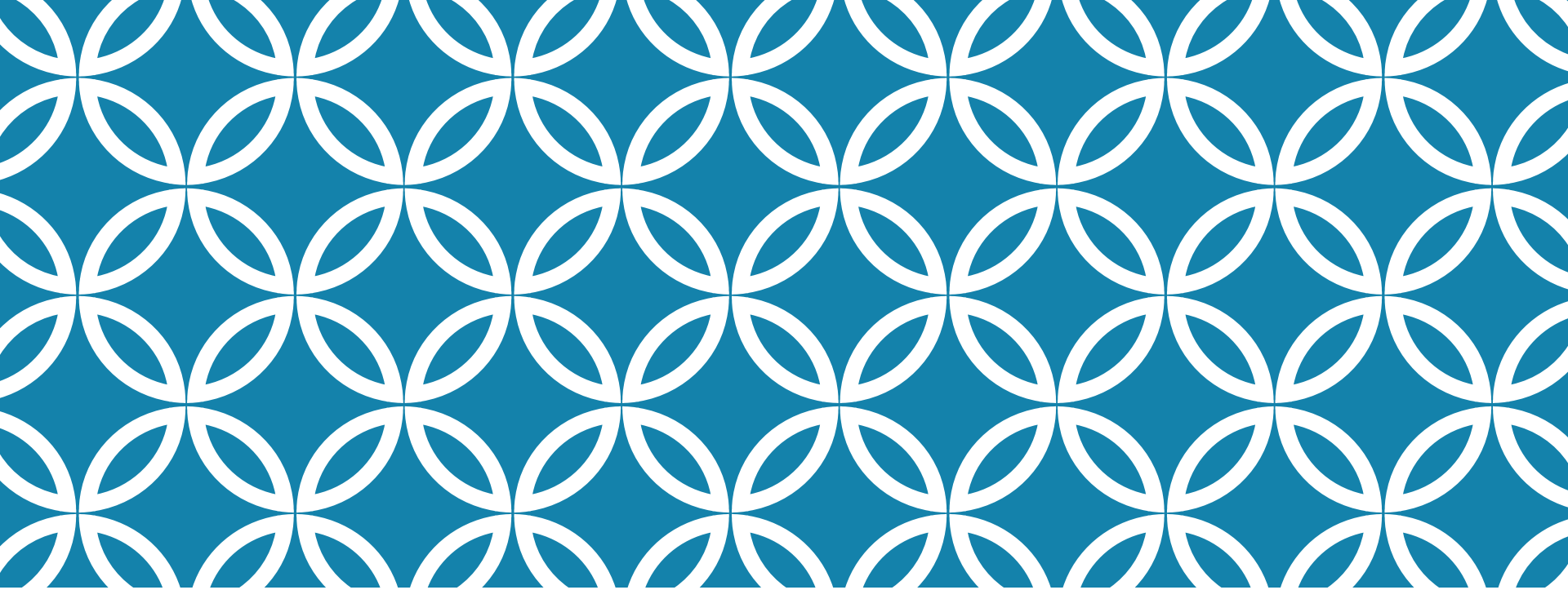


SOEN 691Big Data Analytics Team Project

Dinesh Kumar
Yiming Ren
Yixuan Li
Zhiyi Ding

Introduction

- Main job : Classification with two classes label “normal” and “anomaly”, using two algorithm online and offline (MC-NN and kNN). We are using offline is KNN as the baseline to compare the performance of MC-NN

- Tasks: Algorithms implementation, results analysis and comparison with two classification algorithms - kNN and MC-NN on the Network Intrusion Detection dataset.

- Implemented Classical kNN and data streaming by MC-NN, using PySpark platform include pyspark.streaming libray.

Dataset — Network Intrusion Detection

- **Background**

  Dataset consists of a variety of network intrusions simulated data in a military network environment. It contains 25192 rows of TCP/IP connection data which we used in kNN and MC-NN experiments.

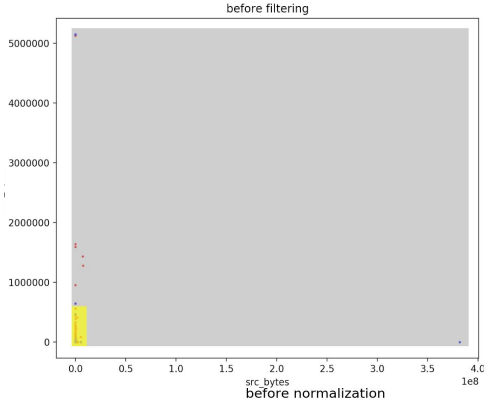
- Features: Each TCP/IP connection contains
  - 3 qualitative features (protocol, service and flag)
  - 38 quantitative features (duration, bytes, rate etc..)
  - 1 class column
- Classes: Anomaly (positive) ／ Normal (negative)
- Data source: https://www.kaggle.com/sampadab17/network-intrusion-detection

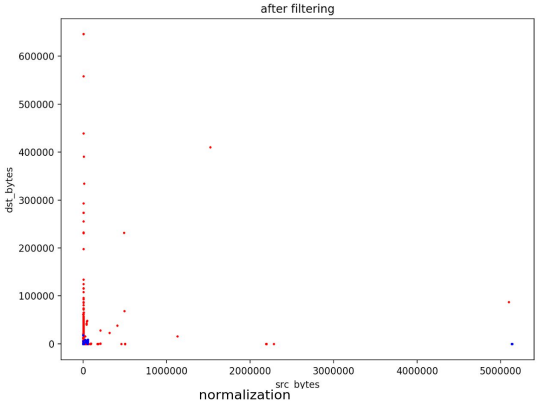
Data preprocessing

During data preprocessing, we did 2 operations on the raw dataset:

1. Outlier detection and filtering

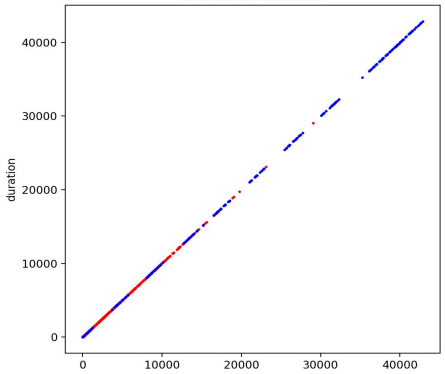
We used an outlier detection method which is called “IQR method” developed by John Tukey. We used the method on 38 quantitative columns.

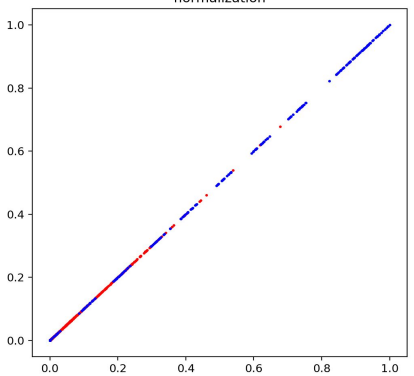




2. Feature normalization

We transformed the 38 quantitative columns to the range of [0, 1]. Then output normalized data to another file.





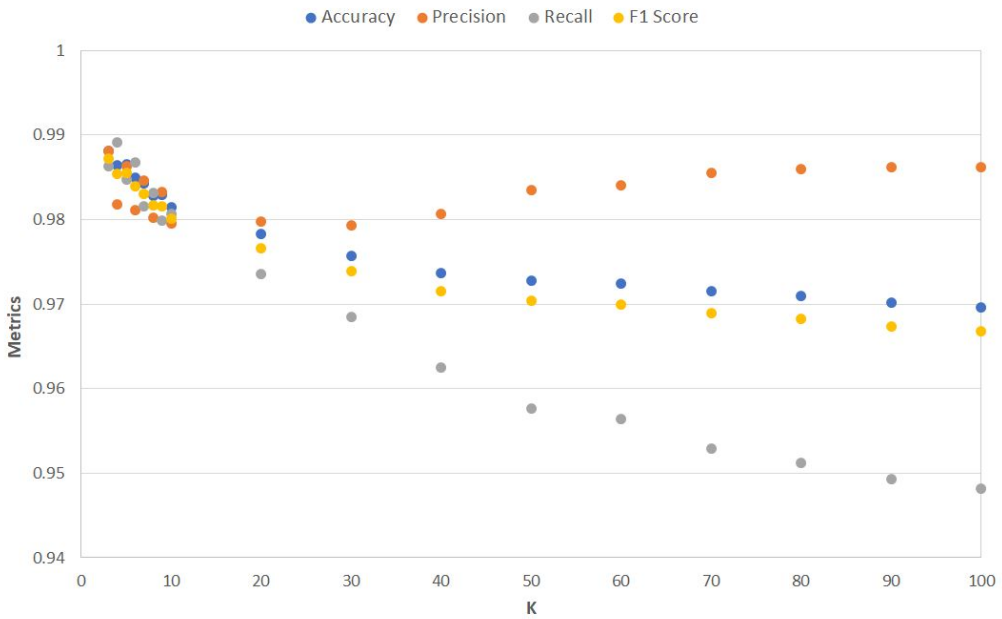
credit: http://colingorrie.github.io/outlier-detection.html

Offline kNN Baseline Experiment

Offline kNN has been evaluated with the entire dataset to provide the baseline for MC-NN evaluation.

- Cross-validation was used to search for hyperparameter K
- Training set and test set ratio: 20% training (~5000 instances), 80% test
- Folds of CV: 5
- Range of K been searched: [3, 9] increment of 1, [10, 100] increment of 10
- Distance of different categorical feature values: 1
- Ran on Compute Canada clusters, thanks to Dr. Tristan Glatard's sponsorship

kNN Results



Averaged evaluation metrics for each K:

- In general, kNN is slow to run but performs very well. All metrics are above 0.94.
- Accuracy, recall and F1 score decrease as K increases.
- Precision first shows a fluctuant decrease when K is lower than 10, then it bounces back as K increases.
- K=3 gives the overall best performance (best accuracy, precision and F1 score, 2nd best recall)

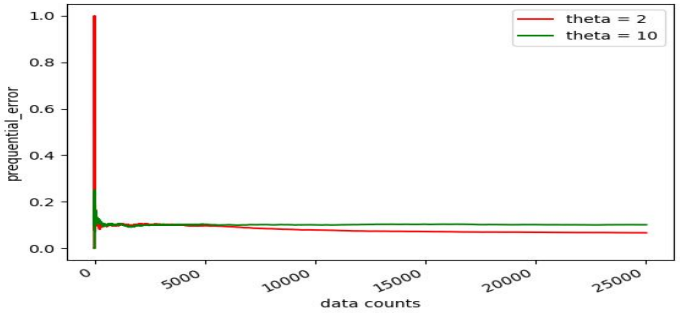
MC-NN Micro-Cluster Nearest Neighbour

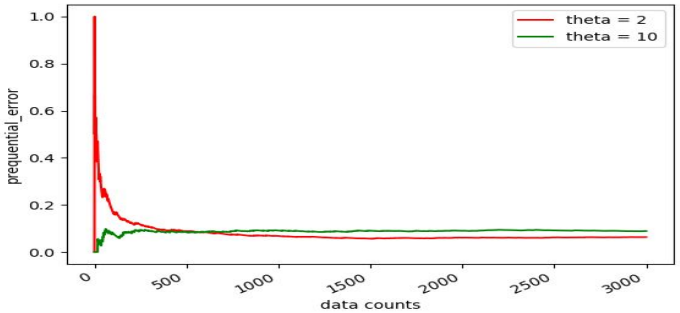
MC-NN is applied to classifier the real-time streaming data. Need to do the classification fast.

- No time to build the training model, save the information of each class as clusters.
- Using the clusters' centroids to classifier the data instances, keep updating as the new instances coming in.
- Using the prequential error to evaluate the performance as for stream learning algorithm. result see next slides
- Set two parameters mx-number of active clusters

  theta-threshold of errors in each cluster

MC-NN Results





- prequential error shows the accumulative miss classification during streaming process.
- As the graph shows, both prequential error started at high level, and gradually became stable to a certain percentage.
- low theta shows the better performance than the high theta.

Comparison KNN with MC-NN

	KNN(k = 3)	MC-NN(theta = 2)
Accuracy	0.988094	0.89131
Precision	0.988112	0.82569
Recall	0.986336	0.97161
F1 Score	0.987218	0.89272

- Overall KNN perform better than MC-NN the entire the dataset based on these four measurements.
- MC-NN is more adaptive to the data streaming process, and run faster.

Questions？