California State Polytechnic University, Pomona

College of Engineering

Electrical and Computer Engineering Department

Senior Project Design

Fall 2019 - Spring 2020

Intrusion Detection System Using Machine Learning

Report

By: Howard Shin

To: Professor Zekeriya Aliyazicioglu

March 8, 2020

Table of Contents

Objective ……………………………………………………………………….……….………..3

Procedure/Outline ..…………………………………………………….……….…….………….3

Summary ...……………………………………………………………….…….…….…………..3

Appendix A …………………………………………………………………….….……………..4

Part 1: Types of Algorithms ……………………………………………………………...4

Part 2: Datasets ……………………..………………………………….……….………..6

Part 3: Python Libraries ………………………………………………………………….7

Appendix B ….……………………………………………………………………….…………..8

Part 1: Initial Testing ..………………….………………………….……………………..8

Part 2: Troubleshooting ……..……….………………………………………...………..11

Appendix C ….……………………………………………………………………….………….12

Part 1: Results ………………….....….………………………………………...………..12

Part 2: Analysis …………………………………………………………………...……..13

References ……………………………………………………...….….……..………...………..15

1.0: Objective

The purpose of the project was to design a Network Intrusion Detection System (NIDS) using different methods of machine learning.

2.0 Procedure

To design the Network Intrusion Detection System, the concept of machine learning was studied. Different algorithms were then analyzed for their purpose. To implement such methods, we chose Python as the programming language due to the existing libraries and ease of use. Along the way, numerous optimizations were made to effectively analyze the datasets. To tackle the design, we chose K-Nearest Neighbor (KNN) Classifier and the Naïve Bayes Classifier. We recorded the accuracy of different methods to determine the effectiveness of each system.

3.0 Conclusion

We compared the effectiveness of KNN Classifier to the Naïve Bayes Classifier on the University of California, Irvine’s KDD Cup 1999 Dataset. The KNN algorithm showed accuracy of 92.59%, while Bayes classifier achieved accuracy of 91.43%. Out of the 311,029 instances, the KNN classifier predicted 3,594 more instances correct than the Bayes classifier. Rather than drastically proving the more effective method, the tests showed that both methods were relatively successful in recognizing the normal network data versus the anomaly data. Throughout the progress of the project, it has been shown that not only is the algorithm important, but also the procedure of communicating with the data also proved to be important.

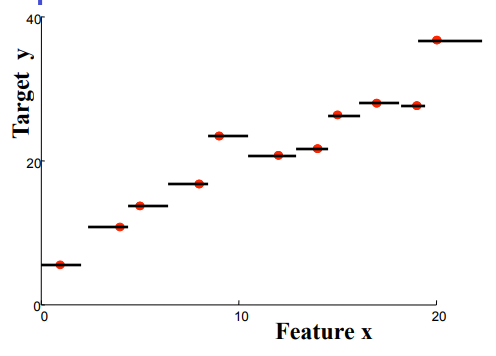
Appendix A

Part 1: Types of Algorithms

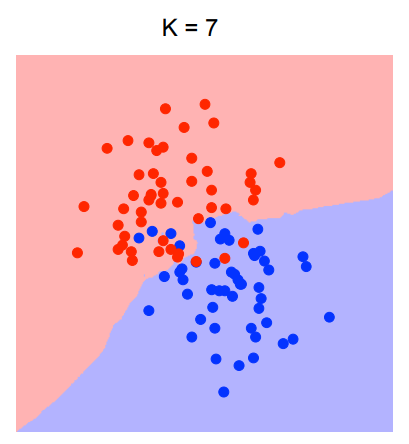
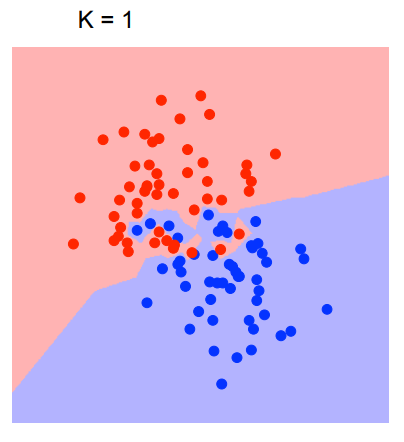
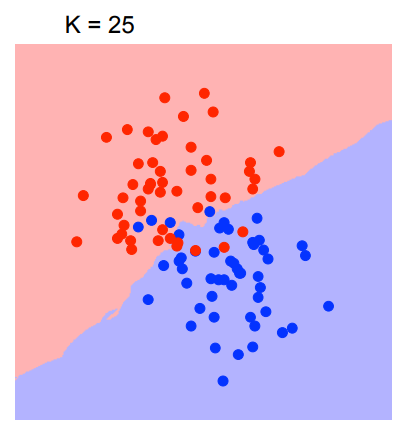
Network Intrusion Detection is a type of supervised classification, meaning a training set is provicded for the machine to train, and new set of data will be assigned with their respective labels after calculations. There are countless approach to supervised classification.

1. **K-Nearest Neighbor:**

KNN algorithm calculates the euclidian distance of a point X to the k amount of closest point around it. Point X is then classified by the set that has the most common feature.

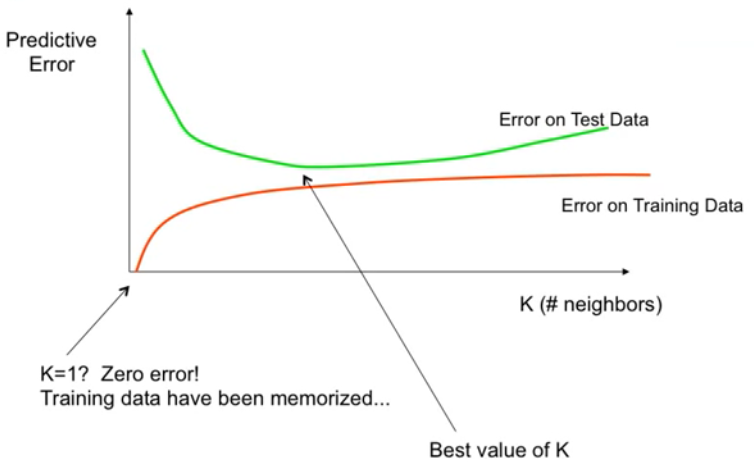


As the k value decreases, the algorithm compares with less neighbors, being more strict with decision making. Therefore, division of the dataset looks more rigid.

Appendix A (continued)

The goal in KNN algorithm is to find the k value in which the error is minimized.



1. **Naïve Bayes classifier:**

Naïve Bayes classifier calculates the probability of each elements and decide the classification based on the highest probability. It follows the following equation:

where,

*P*(*c|x*) is the posterior probability of *class* (c, *target*) given *predictor* (x, *attributes*).

*P*(*x|c*) is the likelihood which is the probability of *predictor* given *class*.

*P*(*c*) is the prior probability of *class*.

*P*(*x*) is the prior probability of *predictor*.

Appendix A (continued)

Part 2: Datasets

When challenging a set of complex algorithms, it was important to choose the correct set of data to use for testing. Couple options were publicly available to use.

1. **KDD Cup 1999:**

KDD Cup 1999 dataset was a set used for The Third International Knowledge Discovery and Data Mining Tools Competition. The database included wide variety of intrusions simulated in a military network environment. The set featured mainly normal connections and 4 different types of attack categories, which are DoS, Probe, R2L and U2R. The attack types further branches to sub-categories. The full dataset contain 4,898,431 instances, along with additional forms of dataset to test.

1. **GureKDDcup:**

The GureKDDcup dataset adds the payload of each connection to the KDD Cup ‘99 set from above. It includes a 9.3 GB size of dataset provided by the MIT, which includes number of samples, attack categories, duplicate records and more.

1. **NSL-KDD:**  
   The NSL-KDD dataset is an improved version of the KDD Cup ‘99 set, where it reduces redundant data, removes any duplicate records, and represent a more realistic data that can be present in the public.

In the end, KDD Cup 1999 was chosen due to its reliable documentation and ample amount of data provided by it.

Appendix A (continued)

Part 2: Python Libraries

To assist with the machine learning, there exists Python libraries built with the purpose of data science.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | ***Tensorflow*** | ***Keras*** | ***scikit-learn (sklearn)*** |
| Pros | -High performance  -Flexible architecture  -Popular | -Built on Tensorflow  -Easy to use  -Well documented | -Easy to utilize with Python’s default libraries.  -Easy to use overall  -Well documented |
| Cons | -Difficult to use  -Difficult to debug | -Difficult to debug | -Does not support GPUs  -Does not run on iOS devices |

As scikit-learn’s cons did not apply to this project, scikit-learn was utilized.

Appendix B

Part 1: Initial Testing

In the KDD Cup 1999 full dataset, there were 4,898,431 instances. In comparison there were 494,021 instances in the 10% subset of the dataset, and 311,029 instances in the corrected dataset. For the project, the 10 percent set was used for training purposes, and the corrected dataset for testing purposes.

One of the problems that came across the dataset was that it contained string data, such as protocol type, service, and etc. As the sklearn library’s machine learning algorithms strictly deal with integers and floats only, it was necessary to convert each label to number. As a rudimentary method of bypassing the problem, only the numbered data were processed, while any data with string were ignored.

To begin, the KNN method was tested. To better evaluate the dataset, it was necessary to normalize the features uniformly, as the dataset included different type of data which contained wide range of their own. With the sklearn’s StandardScaler library, the dataset was scaled. After the process, the K Nearest Neighbor algorithm, with k=5 and default setting, was applied. However, the result was below expectation as it produced 77.38% accuracy.

Appendix B (continued)

To analyze the results, a confusion matrix was produced. The confusion matrix represents the following:

|  |  |
| --- | --- |
| Correct Predictions | Incorrect Predictions |

|  |  |  |
| --- | --- | --- |
|  | Class 1 Prediction | Class 2 Prediction |
| Class 1 Actual | ***True Positive (TP)***: Observation is positive, and is predicted to be positive. | ***False Positive (FP)***: Observation is positive, and is predicted to be negative. |
| Class 2 Actual | ***False Negative (FN)***: Observation is negative, and is predicted to be positive. | ***True Negative (TN)***: Observation is negative, and is predicted to be negative. |

The test produced the resulting confusion matrix with accuracy of 77.38%:

|  |  |  |
| --- | --- | --- |
| n = 311,029 | Predicted: anomaly | Predicted: normal |
| Actual: anomaly | 191,012 | 65,011 |
| Actual: normal | 5,344 | 49,662 |

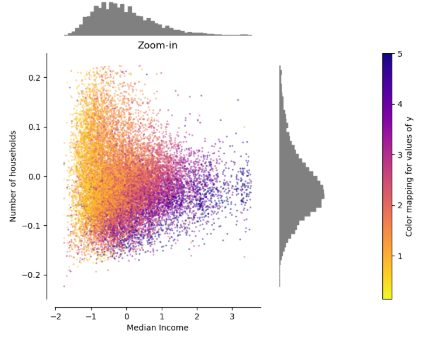
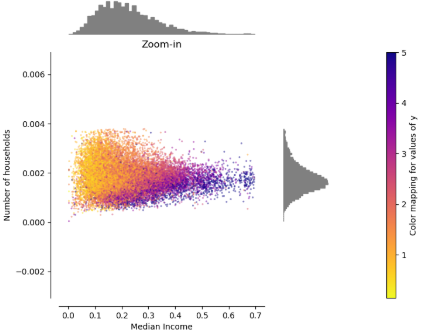
To further experiment, the KNN algorithm was applied with k values ranging from 1 to 39 to check if suboptimal k value was the cause of sizeable inaccuracy.

|  |  |
| --- | --- |
| K Value | Accuracy |
| 1 | 78.08% |
| 2 | 78.30% |
| 3 | 77.49% |
| 4 | 77.58% |
| 5 | 77.38% |
| 6 | 77.67% |
| 7 | 77.48% |
| 8 | 77.59% |
| 9 | 77.48% |
| 10 | 79.60% |
| 11 | 79.56% |
| 12 | 79.83% |
| 13 | 79.76% |
| 14 | 79.82% |
| 15 | 79.77% |
| 16 | 79.87% |
| 17 | 79.81% |
| 18 | 79.99%  The highest accuracy occurred at K = 20, in which the accuracy was 80.03%. This, however, was not enough of a score to be desired. |
| 19 | 79.94% |
| 20 | 80.03% |
| 21 | 78.71% |
| 22 | 78.87% |
| 23 | 78.14% |
| 24 | 78.38% |
| 25 | 78.30% |
| 26 | 78.40% |
| 27 | 78.32% |
| 28 | 78.51% |
| 29 | 78.42% |
| 30 | 78.50% |
| 31 | 78.45% |
| 32 | 78.50% |
| 33 | 78.46% |
| 34 | 78.57% |
| 35 | 78.57% |
| 36 | 78.63% |
| 37 | 78.59% |
| 38 | 78.63% |
| 39 | 78.68% |

Appendix B (continued)

Part 2: Troubleshooting

To combat the inaccuracies, various modifications were done to the structure of the program. With the usage of label encoder, provided by the sklearn library, it was possible to convert the existing string format data, which were initially omitted. Additionally, it was more optimal to use the MinMaxScaler rather than the StandardScaler to normalize the dataset, as StandardScaler has greater bias towards data outliers, and MinMaxScaler normalizes the data to a greater and narrower extent, working better with large dataset.

**Data after standard scaling VS Data after min-max scaling**

In addition to the scaler, the KNN algorithm contains different algorithms within which decides the method of calculation. Rather than using the default settings, which utilizes the “brute” algorithm, the algorithm was manually chosen to the “ball tree” method, which is better designed for a large dataset like KDD Cup 1999 set.

Appendix C (continued)

Part 1: Results  
Applying the newly improved K Nearest Neighbor predictor, it produced the best results at k value of 5, showing a 92.59% accuracy with the following confusion matrix.

|  |  |  |
| --- | --- | --- |
| ***K Nearest Neighbor Classifier (k=5)*** | | |
| n = 311,029 | Predicted: anomaly | Predicted: normal |
| Actual: anomaly | ***TP:*** 227,718 | ***FP:*** 22,718 |
| Actual: normal | ***FN:*** 328 | ***TN:*** 60,265 |

Additionally, the Naïve Bayes classifier was utilized which produced a 91.43% accuracy with the following confusion matrix.

|  |  |  |
| --- | --- | --- |
| ***Naïve Bayes Classifier*** | | |
| n = 311,029 | Predicted: anomaly | Predicted: normal |
| Actual: anomaly | ***TP:*** 225,559 | ***FP:*** 24,877 |
| Actual: normal | ***FN:*** 1,763 | ***TN:*** 58,830 |

However, the accuracy assumed equal costs for both types of errors. Therefore, we calculated precision, recall, f-beta score, and support for each mode to better analyze each data.

Appendix X (continued)

Part 2: Analysis

However, the accuracy assumed equal costs for both types of errors. Therefore, we calculated precision, recall, f-beta score, and support for each mode to better analyze each data.

The *precision* is the ability for the classifier not to label as positive a sample that is negative, calculated by the following equation:

The *recall* is the ability of the classifier to find all the positive samples, calculated by the following equation:

The *f-beta score*, also known as *f1-score*, is the weighted harmonic mean of precision and recall, representing both precision and recall as a function. The f1-score has a worst value of 0 and best value of 1. It is calculated by the following equation:

The *support* represents the number of true labels for each instance.

The *macro average* takes the metrics for each label and finds their unweighted mean. This does not take label imbalance into account. In contrast, the *weighted average* calculates metrics for each label, and finds their average weighted by support. This alters ‘macro’ to account for label imbalance; it can result in an F-score that is not between precision and recall.

Appendix C (continued)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***K Nearest Neighbor Classifier (k=5)*** | | | | |
|  | precision | recall | f1-score | support |
| anomaly | 1.00 | 0.91 | 0.95 | 250436 |
| normal | 0.73 | 0.99 | 0.84 | 60593 |
| accuracy | X | X | 0.93 | 311029 |
| macro avg | 0.86 | 0.95 | 0.90 | 311029 |
| weighted avg | 0.95 | 0.93 | 0.93 | 311029 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Naïve Bayes Classifier*** | | | | |
|  | precision | recall | f1-score | support |
| anomaly | 0.99 | 0.90 | 0.94 | 250436 |
| normal | 0.70 | 0.97 | 0.82 | 60593 |
| accuracy | X | X | 0.91 | 311029 |
| macro avg | 0.85 | 0.94 | 0.88 | 311029 |
| weighted avg | 0.94 | 0.91 | 0.92 | 311029 |

Although not by a lot, the KNN classifier exceeded the Naïve Bayes classifier in nearly every category, proving it to be more effective in predicting the KDD Cup 1999 dataset. However, the differences are also subjectively minimal enough for the Bayes classifier to be considered successful model.

References

Chaudhari, Rashmi Ravindra, and Sonal Pramod Patil. “INTRUSION DETECTION SYSTEM: CLASSIFICATION, TECHNIQUES AND DATASETS TO IMPLEMENT.” International Research Journal of Engineering and Technology, vol. 04, no. 02, Feb. 2017, pp. 1860–1866., <https://www.irjet.net/archives/V4/i2/IRJET-V4I2366.pdf>.

Chandrashekhar, A. M., and Chandana P. “COMPARISON OF DATA MINING TECHNIQUES USED IN ANOMALY BASED IDS .” International Research Journal of Engineering and Technology, vol. 04, no. 06, June 2017, pp. 1210–1215., <https://www.irjet.net/archives/V4/i6/IRJET-V4I6223.pdf>.

Kristiansen, Søren L. “Nearest Neighbors with Keras and CoreML.” Medium, Medium, 4 Oct. 2018, [medium.com/@sorenlind/nearest-neighbors-with-keras-and-coreml-755e76fedf36](mailto:medium.com/@sorenlind/nearest-neighbors-with-keras-and-coreml-755e76fedf36).

“Compare the Effect of Different Scalers on Data with Outliers.” Scikit, scikit-learn.org/stable/auto\_examples/preprocessing/plot\_all\_scaling.html.

“NSL-KDD Dataset.” University of New Brunswick, [www.unb.ca/cic/datasets/nsl.html](http://www.unb.ca/cic/datasets/nsl.html).