## 1 PROBLEM STATEMENT

To develop a generic model for the cyber security domain because all the machine learning models in this domain have been abstracted as per industry regulations.

## 2 PROBLEM DESCRIPTION

The cyber security domain has a few implementation for machine learning to be used but they have been abstracted because of the industry regulations. Thus here we try to develop a generic model that works for all datasets in the cyber security domain. The data sets of NVD will be used and the models developed and tested against the CVSS standard .

## 3 LITERARY SURVEY

Since the domain we have chosen is cyber security that is implemented in the corporate sector all the research work has just the basic ideation and the implementation has been abstracted out due to the confidentiality that is required. The gist that they provide is to get the Lagrangian model to remove the bias in the datasets. An IEEE paper on the same domain was taken as the base. The changes we have intended to implement are to make the algorithm be able to work on any kind of biased data sets in the cyber security domain.

# 4 INTRODUCTION

The datasets from the industry were hard to find and so we used the NVD provided by the U.S. Department of Homeland Security's Computer Emergency Readiness Team (US-CERT). This had to be used to train a k-means model that is generic enough to support additional parameters. And finally the model would be tested on the golden standard of the Common Vulnerability Scoring System (CVSS) v3.0 standards.

# 5 WORKFLOW

We will be using the NVD dataset as the base and create the labels for the records so that they could be used for the training process. Next, it would be necessary to get the most important features that would be required to find the anomalies with greater accuracy. Finally, we get to choose the best model required for the use case at hand. This model was tested against the Common Vulnerability Scoring System (CVSS) v3.0 standards as benchmark.

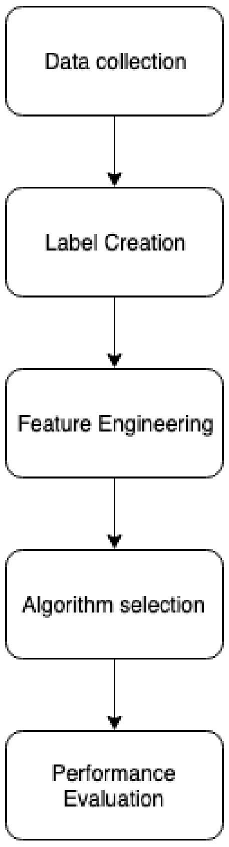

**Figure 5.1 General outline of workflow**

# 6 SYSTEM ARCHITECTURE

The below figure gives an overall idea of how the components work together with the pipeline complete. Let us look into detail into all the modules.

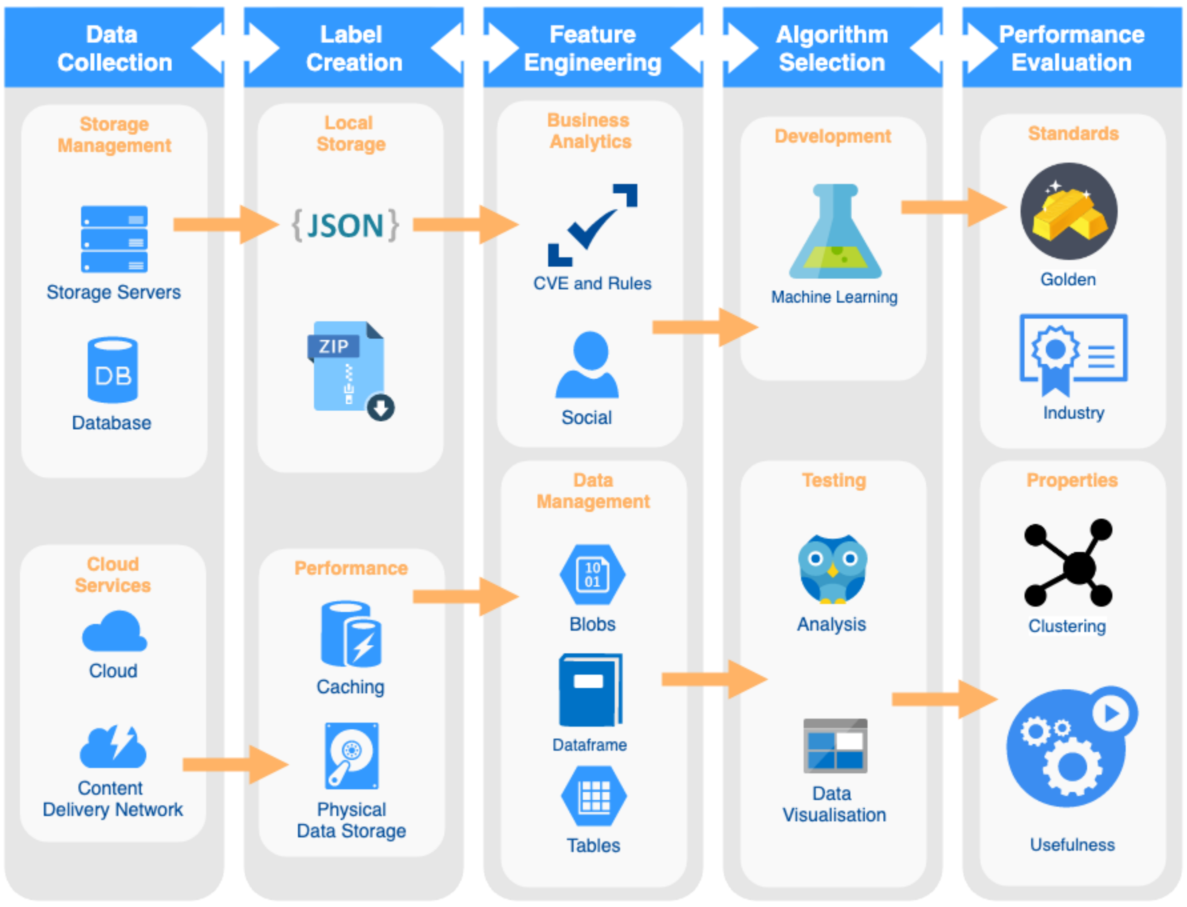

**Figure 6.1 Detailed System Architecture**

## 6.1 DATA COLLECTION

The data set provided by the National Vulnerability Database maintained by the U.S. Department of Homeland Security's Computer Emergency Readiness Team (US-CERT) has been used. We used python to build all the modules. These data sets were provided at different endpoints by the organisation for the CDN to reduce the server load. Packages in python were used to hit the server and concatenate the data into a JSON file.

## 6.2 LABEL CREATION

The data was accumulated as JSON files in accordance to the year of data creation. And as the file size was a bit large we zipped files for later use and loaded them into our local hard disks. The JSON parser module in Python used a custom caching mechanism to provide faster access to the values of the required fields. This component was provided by Pandas and so we stuck to using the same for developing our machine learning models as well.

## 6.3 FEATURE ENGINEERING

The models had been evaluated based on the Common Vulnerability Scoring System (CVSS) v3.0 standards which is commonly followed in the cyber security domain. But to make the model more generic we also introduced a few parameters for the social factors that would depend on the use case domain. All the input was stored as blobs within dataframes, which in turn were retrieved from the tables that pandas provides.

## 6.4 ALGORITHM SELECTION

The models had to be developed on the Lagrangian bias removed data sets and had work in all domains like spam calls, phishing emails and inappropriate conversations in social media. As a result they had to be tested against the same CVSS standard. As the output is scalar as with the CVSS standard we could go with the k-means models as they would be efficient and generic enough.

## 6.5 PERFORMANCE EVALUATION

To test the models the golden standard of Common Vulnerability Scoring System (CVSS) v3.0 standards was used. This gives a score from 0 to 10 based on the ease to exploit the vulnerability and the industry impact it will be having. Thus the above will act as the gold standard. We will also check the information gain from the clustering of the data points and try to arrive at conclusions.

## 7 IMPLEMENTATION

The implementation was done using Python as it provided all the functionalities required by the project. The Python shell was run using a Jupyter notebook as it has the capabilities of the shell that Python provides.

## 7.1 DATA LOADING

The data loading was done using Pandas and NumPy which are very useful for data loading, cleaning and analysis. First, we had to change the default number of rows and columns that a Jupyter notebook allows to be displayed for easier analysis of the data. We also had to implement a cutoff date, as it would allow us to check the results with the major security incident of Equifax which was a data breach that exposed the personal information of around 150 million people. The JSON files had to be accessed from https://nvd.nist.gov which is the official site that provides the dataset. Once the files had been downloaded and extracted, they had to be flattened for easier access.

## 7.2 CLEANING

The flattened data had some redundant columns which were not needed, such as the index and a nested JSON column. The "CVE_ID" had to be reordered as they had been arranged in accordance to the severity of the vulnerability rather than the time of finding it out. The column "impact.baseMetricV2.acInsufInfo" has so many NULL values had to be removed as it would provide no additional information. Even after that column was removed a few NULL values remained and on close inspection was found that all contained the word "reject" as the value for the "cve.description.description_data" column.

A sample value in the "cve.description.description_data" column.

```
[{'lang': 'en',
 'value': '** REJECT ** DO NOT USE THIS CANDIDATE NUMBER. ConsultIDs:
none. Reason: The CNA or individual who requested this candidate did not associate it
with any vulnerability during 2016. Notes: none.'}]
```

Drilling down into the "cve.problemtype.problemtype_data" column, we found that the CWE codes were embedded within JSON objects and had to further flatten the JSON file for getting the primary and secondary CWE codes. When having a look at the CWE type there were several entries for what is essentially the same thing (unknown CWE type), which was converted from 'NVD-CWE-Other' and 'NVD-CWE-noinfo' to 'UNKNOWN'.

## 7.3 WEB SCRAPING

Once the preprocessing was done, we had to understand what the codes mean, but there was no database to get that information either as an API(Application Programming Interface) or a JSON file. The NVD does, however, provide a structured table on its website, which we scraped using BeautifulSoup library for Python. The details about the codes were provided in the link https://nvd.nist.gov/vuln/categories. Then this description was added to the columns list and the redundant "Name" column was removed.

## 7.4 CWE CODE ANALYSIS

For the data analysis part, matplotlib was used for creating the data-rich graphics. Now that the various codes had been consolidated with their definitions into one data frame, we could begin analyzing the information we had, both quantitatively and graphically. To start, a simplelist all of the CWE codes by frequency was done, in descending order.

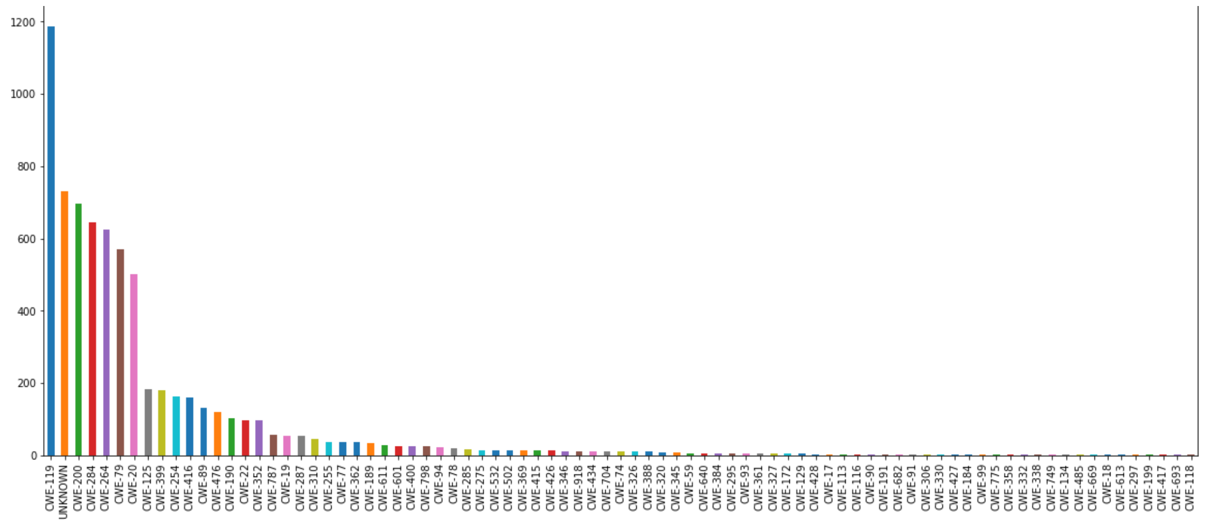

**Figure 7.1 Primary CWE Code by Incidence in 2017 NVD Data**

As a few of the CWE Codes were very common, we had to look into a few. As a first step, we examined the most common primary code, CWE-119, and its description.

CWE-119
Buffer Errors
The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer.

The same was done for the secondary codes and the results were obtained as follows.

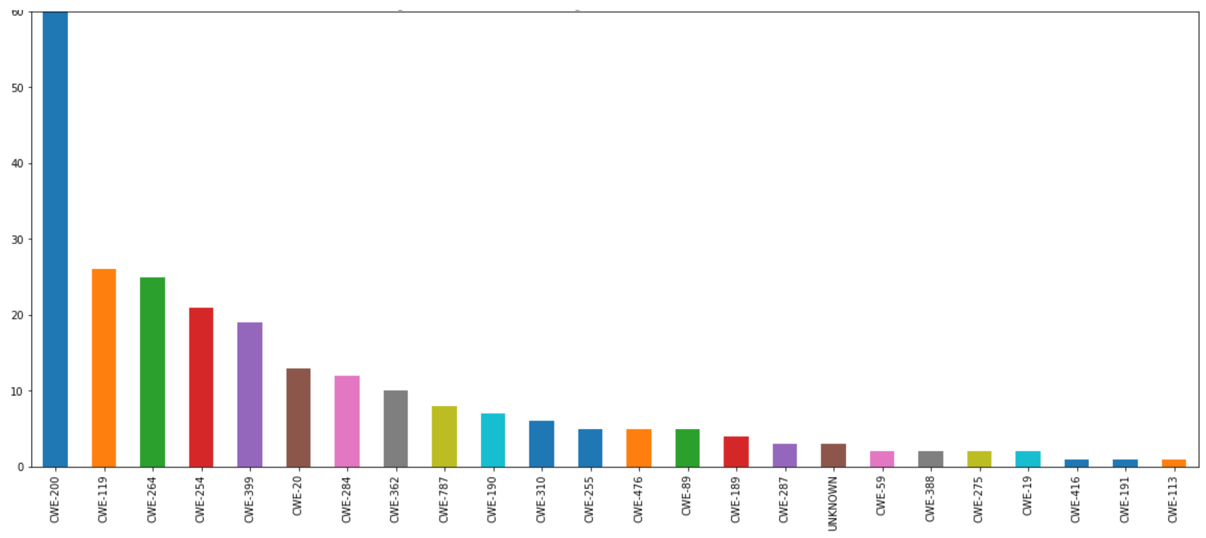

**Figure 7.2 Secondary CWE Code by Incidence in 2017 NVD Data**

## 7.5 CVSS SCORE MAPPING

The fact that an information leak / disclosure is the most common companion when another vulnerability has been discovered seems quite logical. In order to further examine the data, we drew a graph for a variety of different NVD characteristics in order to draw out any potential trends. We used the CVSS 3.0 score to quantify the severity of the various vulnerabilities in the data. In the final cleaned data frame (prior to creating dummy variables), the CVSS 3.0 score is found in the "impact.baseMetricV3.cvssV3.baseScore" column. Then we had to sort the incidents by primary CWE code and analyze the mean CVSS 3.0 score, to identify the most dangerous cybersecurity vulnerabilities. To begin, we plotted the distribution of CVSS scores, and found that they are not normally distributed, but are rather left-skewed. This makes it somewhat difficult to identify whether a score is statistically significant or not.

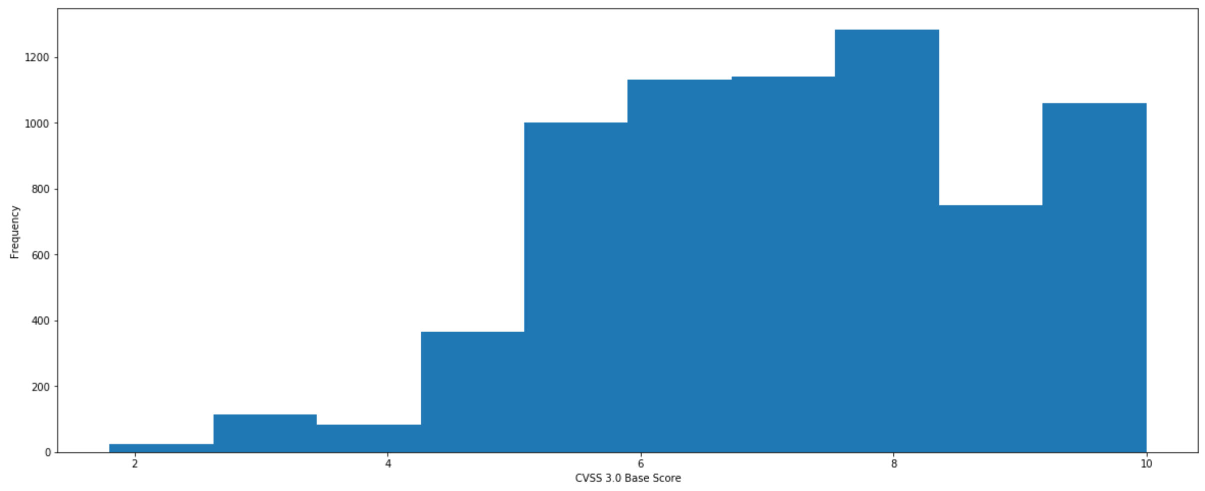

**Figure 7.3 Distribution of CVSS 3.0 Base Score in 2017 NVD Data**

Below is a plot of the frequency of a primary CWE code against the mean base CVSS 3.0 score associated with it. It is hard to identify any trends, because the mean CVSS score fluctuates so much, and as one moves to the far right of the x-axis, the score is potentially just the results of outliers having a huge impact because of the small sample size.

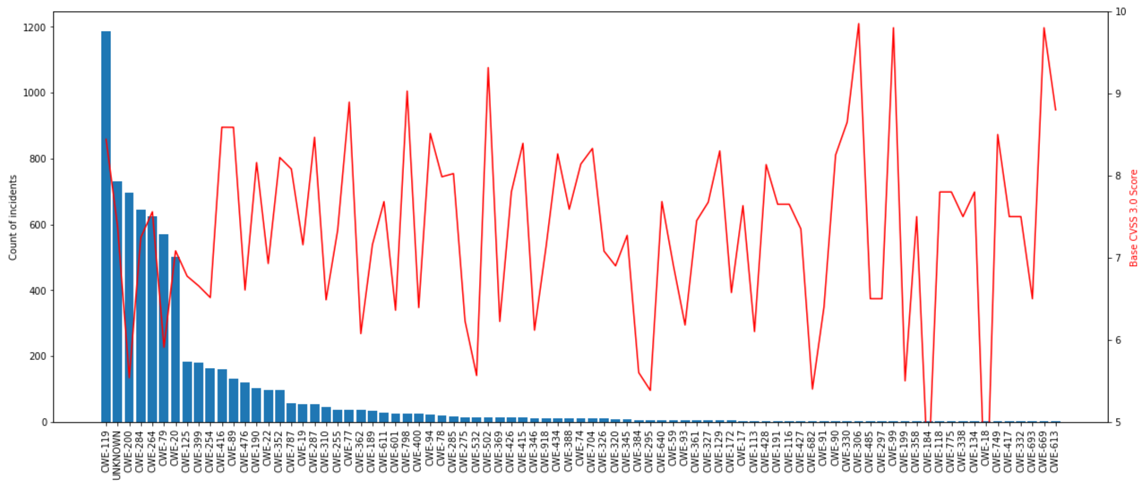

**Figure 7.4 Primary CWE Code**

## 7.6 UNSUPERVISED LEARNING

We are going to use an unsupervised learning technique called k-means clustering in an effort to identify groups of vulnerabilities in the database that might have common characteristics. An excellent summary of the mathematical basis for k-means clustering is available from the paper "An Efficient k-Means Clustering Algorithm: Analysis and Implementation," by Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and

Angela Y. Wu: "In k-means clustering, we are given a set of n data points in d-dimensional space R^d and an integer k and the problem is to determine a set of k points in Rd, called centers, so as to minimize the mean squared distance from each data point to its nearest center." In the words of two other researchers, however, k-means clustering is "sensitive to noisy data and outliers." Extreme values for a small number of data points can thus have an outsized impact on the location of a cluster's centroid in k-means clustering, and we want to ensure that the clusters are not abnormally distorted. Thankfully, in the case of NVD data, this is not a problem. Although we have seen that the CVSS 3.0 base scores are not normally distributed, they are partially normalized due to the fact that they are ranked on a 0-10 scale. Furthermore, each incident has a small (3-4 value) range of possible different categorical values. Because of these characteristics of the dataset, the NVD is thus a good candidate for analysis using k-means clustering. In order to implement this algorithm practically, we used scikit-learn, a common machine learning library for Python, specifically using the "KMeans" function. The silhouette score peaks with 2 and 6 clusters. Since we were trying to achieve the greatest analytical fidelity while still providing actionable results, we selected 6 clusters for the analysis.

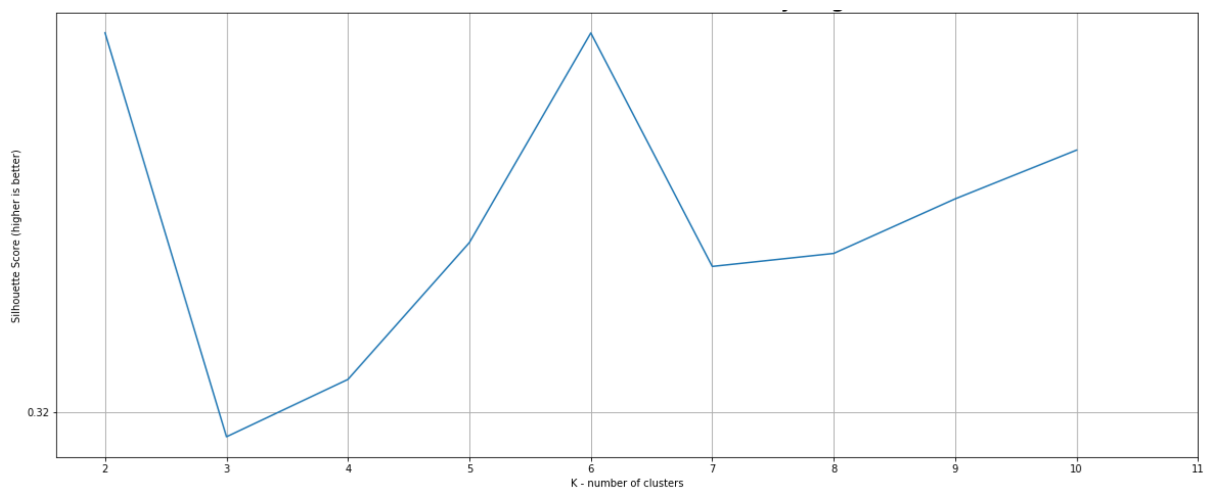

**Figure 7.5 Usefulness of Various Cluster Numbers in Analyzing NVD Data**

## 7.7 PERFORMANCE ANALYSIS

For the performance analysis of the k-means model, we find the mapping between the cluster and its score in CVSS. Cluster 2 appears to include substantially more points than all of the other clusters, interestingly. We then joined the predicted cluster of each incident back onto the original cleaned data frame, and then evaluated the mean CVSS 3.0 score for each cluster. Cluster 5 has the highest mean CVSS 3.0 base score, and thus contains the overall most dangerous cybersecurity vulnerabilities. Drilling down further, however, we plotted the NVD entries in a 2-dimensional space, with the x-axis representing the vulnerabilities CVSS 3.0

exploitability score, and the y-axis representing its equivalent impact score. The highest-risk vulnerabilities are those with the greatest impact and exploitability, thus should pay special attention to the top right corner of the graph. We further color coded each point by cluster to determine if we can detect any useful trends from the analysis we performed previously.

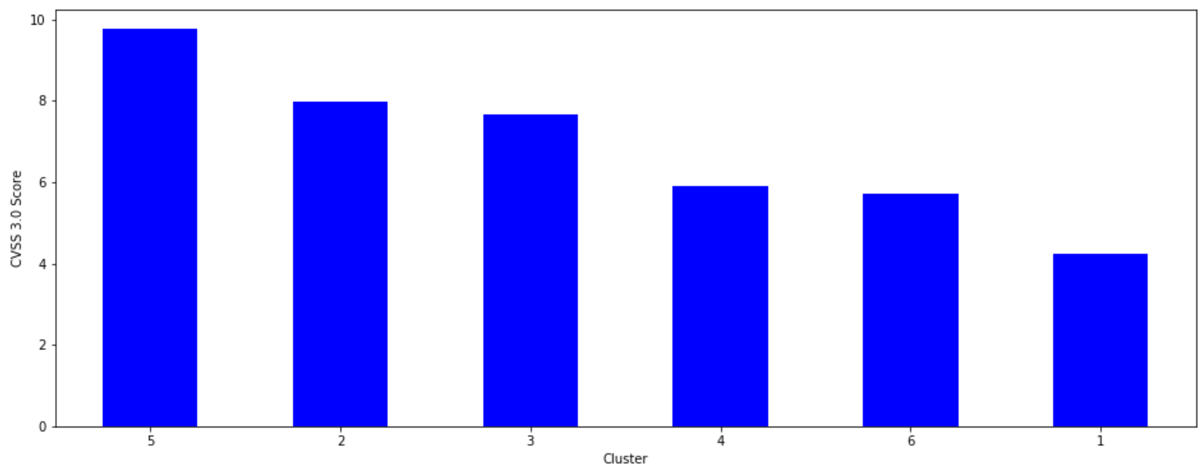

**Figure 7.6 CVSS 3.0 score for Clusters**

It is important to note that the clustering was done based more than just the CVSS 3.0 impact and exploitability scores; the algorithm incorporates all of the numerical and categorical features found in the feature_analysis data frame. Since it is difficult to visualize more than two dimensions, however, we used the  impact and exploitability as the two relevant axes.

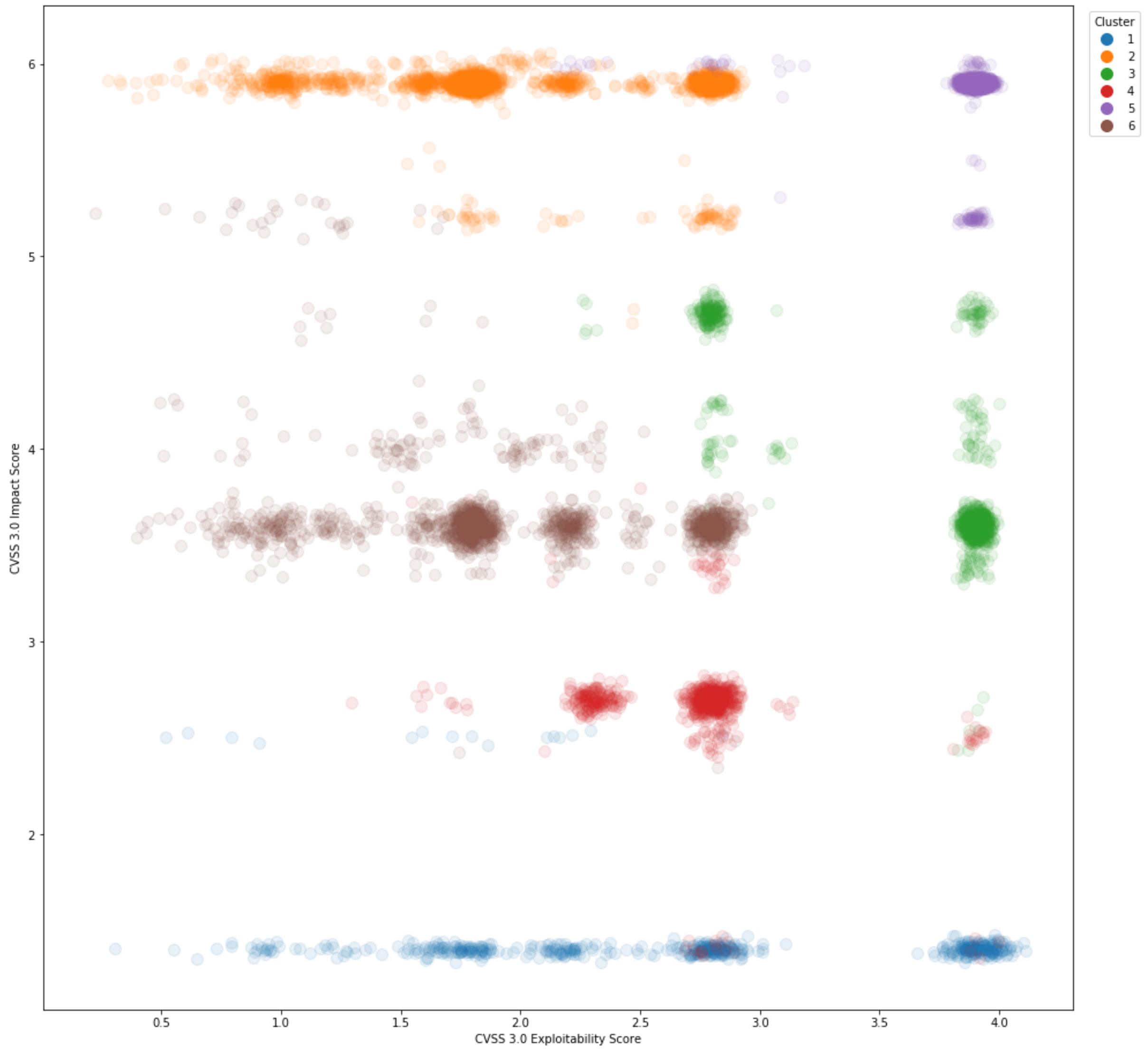

**Figure 7.7 CVSS 3.0 impact and exploitability score**

The unsupervised learning algorithm seems to have done its job, creating several distinct groups of vulnerabilities which are easily distinguishable by their relative impact and exploitability.

## 7.8 CVE-2017-5638

Since we had mentioned the Equifax hack as a prime example of the dangers of leaving vulnerabilities unaddressed, we thought to conclude this project by examining the actual NVD incident responsible for the breach. According to press reporting, it was by exploiting CVE-2017-5638 that the hackers infiltrated Equifax's networks. It seems that the Equifax vulnerability, even though it has a less-frequently appearing CWE code, was in fact a member of cluster 5, which in our model previously identified as containing the most dangerous cybersecurity vulnerabilities. Additionally, simply plotting of the vulnerability used in the Equifax hack is quite revealing and demonstrates how analysis of even a simple two-dimensional model should have triggered the company's security team to take more decisive action upon learning about the existence of a vulnerability in a public US-CERT release from March 2017.

## 8 CONCLUSION

More generally, this project confirms quite a few commonly-held cybersecurity "rules of thumb." Primarily, many of the most impactful and exploitable vulnerabilities are quite common and easy for hackers to employ, especially buffer overflows, SQL injections, and poor credential management. Furthermore, as the k-means cluster analysis has shown, these vulnerabilities share some common characteristics with each other. Addressing the known universe of vulnerabilities will thus require greater discipline on the part of individual users, engineers, and leaders in order to identify and close the gaps through which hackers are most likely to enter computer networks. Renewed emphasis on this "low-hanging fruit" will likely allow for a greater return on investment for future cybersecurity efforts rather than dedicating resources towards addressing more complex but rare software flaws.