Predict CVSS score for new CVEs

Rudra Panda

Abstract

This article demonstrates the use of text analysis and linear regression to predict score for a newly identified software vulnerability. The solution demonstrated here relies on the historical CVE (Common Vulnerability Enumeration) data stored in NVD (National Vulnerability) database, and scikit-learn libraries. It uses the "summary" text of the vulnerability to extract features, and then predicts CVSS score (both V2 and V3 scores).

The RMSE (root mean squared error) for the current experiment, which measures how close are the predictions to the actuals, is 1.33.

I. Introduction

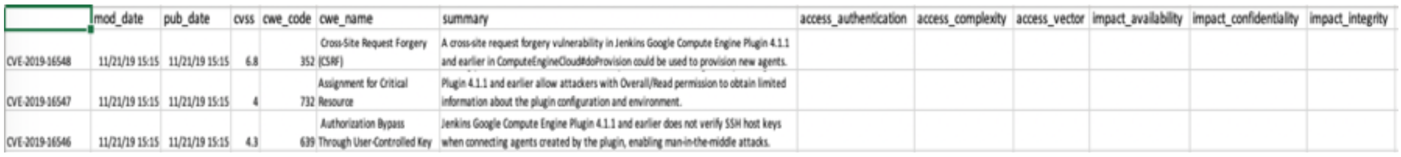
CVSS (common vulnerability system) is a vulnerability scoring system for software vulnerabilities that has been developed to help software industry to prioritize and remediate vulnerabilities promptly and reduce the number of attacks on systems. There are two versions: V2.0 and V3.x standards. The scores range from 0.0 to 10.0, and their qualitative severity rankings differ from V2 to V3. For V2, Low is 0.0 - 3.9, Medium is 4.0 - 6.9, and High is 7.0 - 10.0. For V3, Low is 0.1 - 3.9, Medium is 4.0 - 6.9, High is 7.0 - 8.9, and Critical is 9.0 - 10.0.

After disclosure of a vulnerability, the CVSS score may not be available for few days. In that situation, the proposed solution can be used to get an estimation of possible score for the newly identified vulnerability.

The rest of the paper describes data, its source, approach for extracting features, and use of LinerRegression to predict scores for different sets of features.

II. Data

Figure 1 gives a pictorial view of the dataset. The columns used for this experiment are "summary" and "cvss".



	mod_date	pub_date	cvss	cwe_code	cwe_name	summary	access_authentication	access_complexity	access_vector	impact_availability	impact_confidentiality	impact_integrity
CVE-2019-16548	11/21/19 15:15	11/21/19 15:15	6.8	352	Cross-Site Request Forgery (CSRF)	A cross-site request forgery vulnerability in Jenkins Google Compute Engine Plugin 4.1.1 and earlier in ComputeEngineCloud#doProvision could be used to provision new agents.						
CVE-2019-16547	11/21/19 15:15	11/21/19 15:15	4	732	Assignment for Critical Resource	Plugin 4.1.1 and earlier allow attackers with Overall/Read permission to obtain limited information about the plugin configuration and environment.						
CVE-2019-16546	11/21/19 15:15	11/21/19 15:15	4.3	639	Authorization Bypass Through User-Controlled Key	Jenkins Google Compute Engine Plugin 4.1.1 and earlier does not verify SSH host keys when connecting agents created by the plugin, enabling man-in-the-middle attacks.						

Figure 1 CVE Data

This dataset was available on Kaggle. It had 89,660 records. It had CVE data published between Jan, 1999 and Nov, 2019.

III. Feature Extraction and Selection

The summary column is textual data that provides a high-level summary of a CVE when it gets disclosed for the first time. I broke down this text into a bag of words using "CountVectorizer" function available in "sklearn". Here is the code extract

```
# will convert text to number vector
vectorizer = CountVectorizer(stop_words=stop_words)
# convert text into word features
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

For the dataset used, there were 82,934 words after eliminating common English stop words. I used the "SelectKBest" function from "sklearn" to identify the most influential features. I used "f_regression" as the score function that uses F value to determine statistically significant features. Figure-2 shows the relationship between actual and predicted values for different number of features (words) selected.

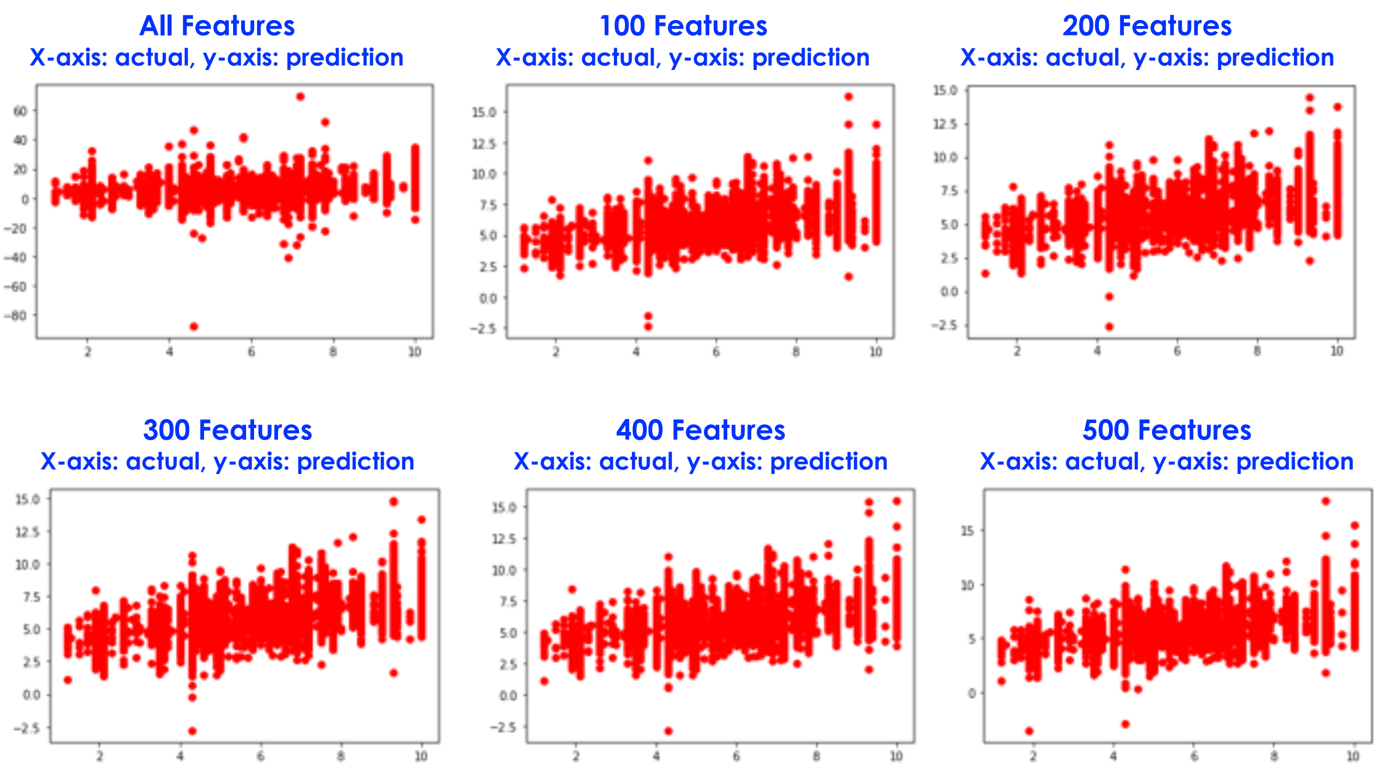


Figure 2 Actual vs. Predicted CVSS Scores

As can be seen from the picture above, there was large variation between prediction and actual when all features (words) were used. Then I experimented with multiple of 100 features (words) at a time. The variation improved with 100 words, 200, and 300 words, and started staying the same or diminished as the number of features (words) were increased.

IV. Model Training

The goal was to predict CVSS score for the newly identified vulnerability that would fall between 0 and 10. Since this was a continuous target, I used multivariate regression for building the model.

For model training, I divided the total dataset into 70 and 30 ratios for training and test data. I used "LinearRegression" from "sklearn" to create the model. I evaluated the model using all, 100, 200, 300, 400, and 500 features. Figure-3 shows the comparison of RMSE (Root Mean Squared Error) and R-square values between these different experiments.

# of Features	RMSE[1]▼	R-Squared[2]
300	1.3285	0.5387
500	1.3309	0.5371
400	1.3403	0.5305
200	1.3474	0.5255

[1] Root Mean Squared Error (RMSE): The mean squared error tells you how close a regression line is to a set of points. The lower is better.

[2] Coefficient Determination (R-squared): The coefficient of determination effectively compares the variance in the data to the variance in the residual. The residual is the difference between the predicted and observed value and its variance is the sum of squares of this difference. The closer to 1 is better.

100	1.3659	0.5124
all	3.3321	-1.9018

Figure 3 Model Scores

V. Model Prediction

I gathered 21 CVEs from NVD site those were latest CVEs to be scored as of 18th Nov 2020. I used the model to predict CVSS scores for these vulnerabilities and compared those scores against the actual scores given by NVD. Figure-4 shows the comparison for V2 scores, and Figure-5 shows for V3.x scores. The x-axis represents the difference between predicted and actual scores. The y-axis represents the frequency of this difference.

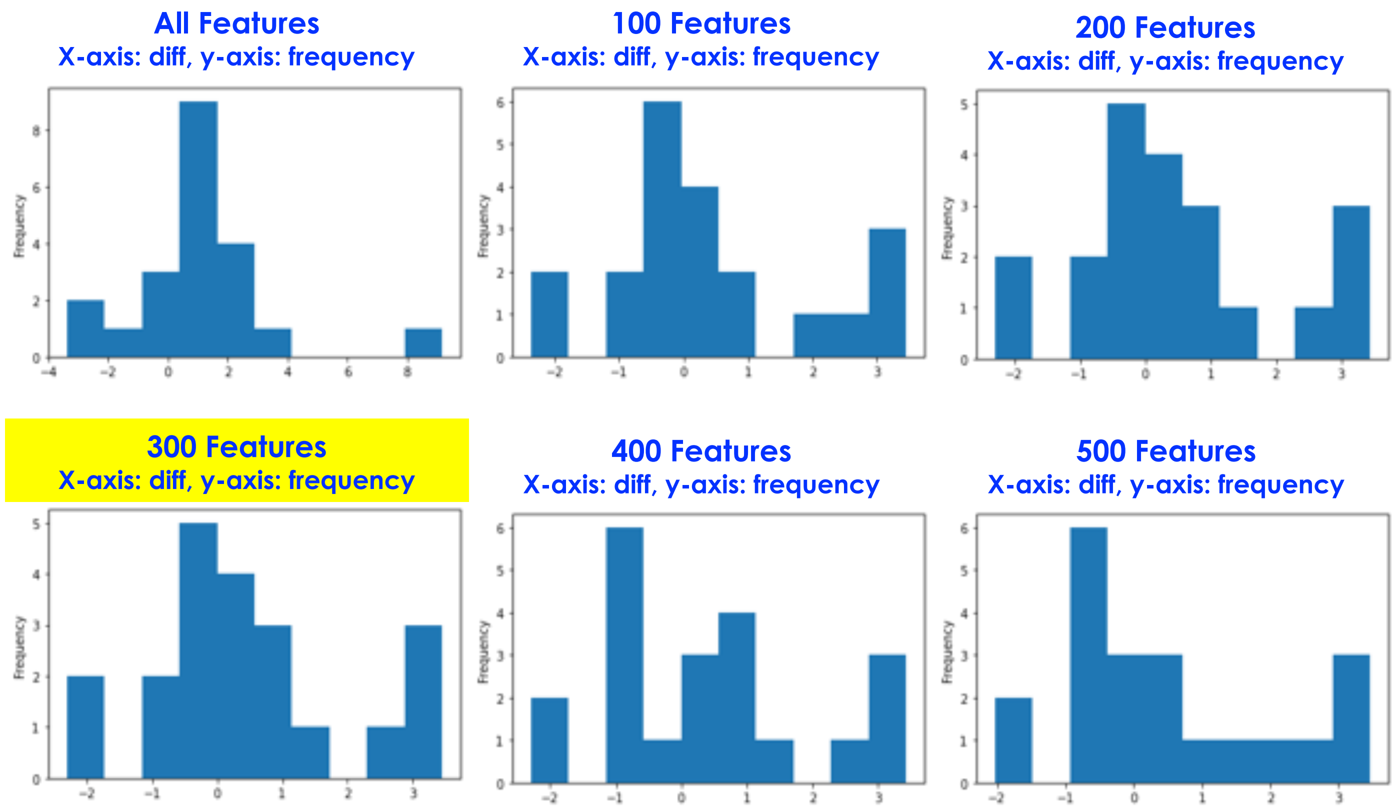


Figure 4 CVSS V2.0 (Prediction - Actual) Distribution

The greater number of data closer to zero on x-axis is better, meaning the model is predicting more CVEs closer to their actual values. As can be seen from the figure above, the experiment with 300 features provides better prediction. However, this is not the case for CVSS V3.x prediction (Figure 5). The possible explanation is that CVSS V3.x score started in Sept, 2019 whereas the dataset used for this experiment has CVSS V2.0 scores only.

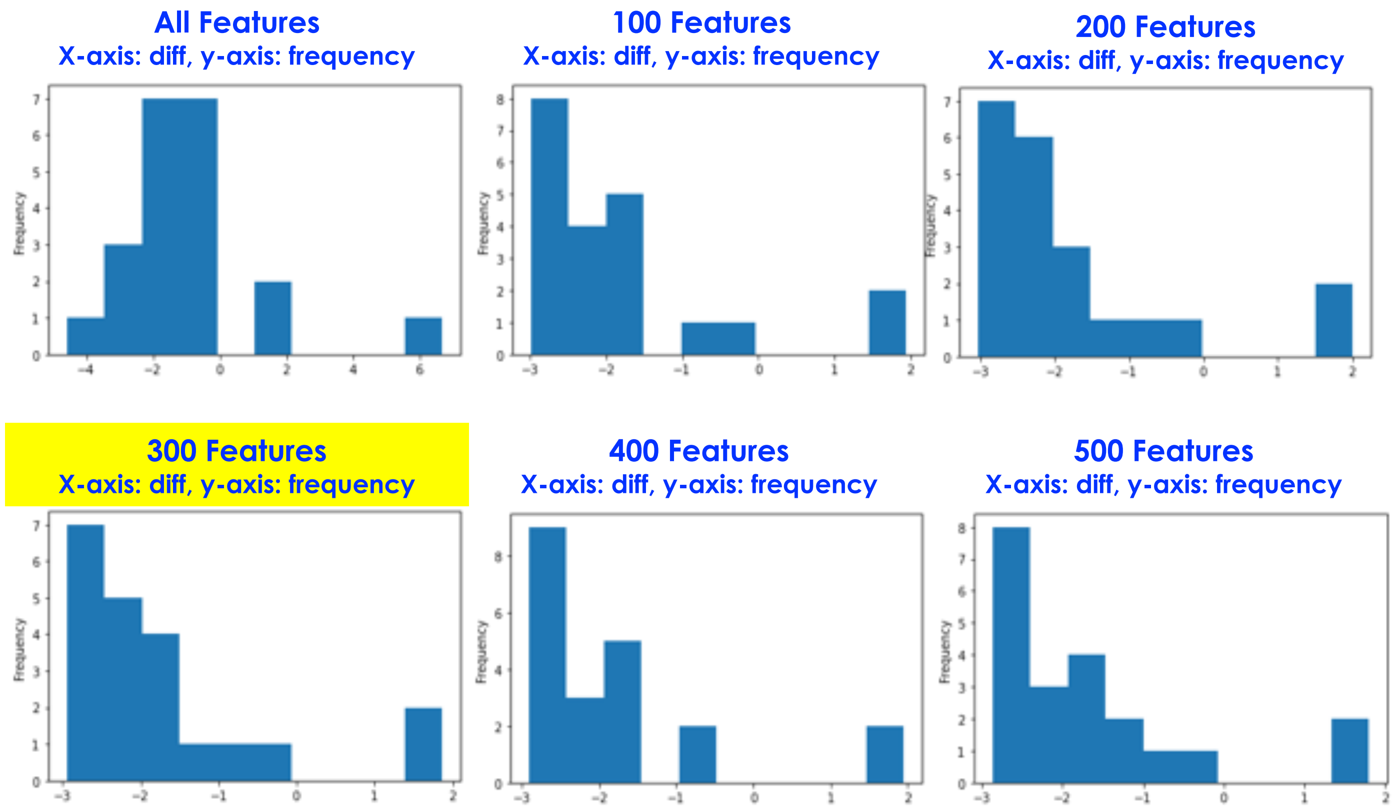
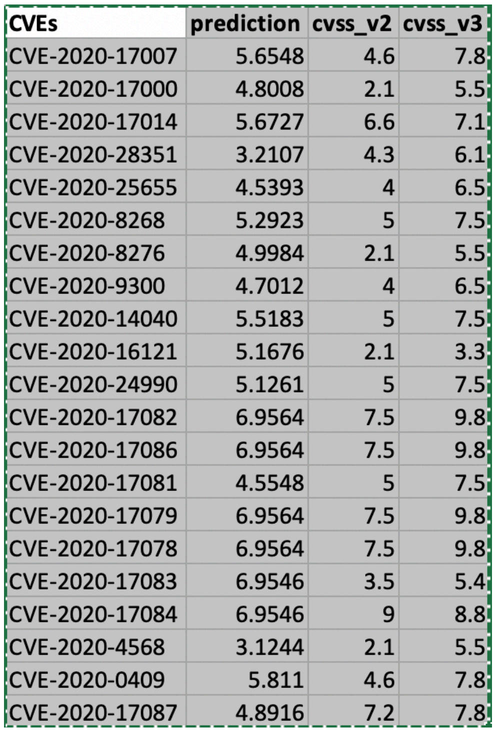


Figure 5 CVSS V3.x (Prediction - Actual) Distribution

Figure 6 shows a comparison between the predicted and actual values.



CVEs	prediction	cvss_v2	cvss_v3
CVE-2020-17007	5.6548	4.6	7.8
CVE-2020-17000	4.8008	2.1	5.5
CVE-2020-17014	5.6727	6.6	7.1
CVE-2020-28351	3.2107	4.3	6.1
CVE-2020-25655	4.5393	4	6.5
CVE-2020-8268	5.2923	5	7.5
CVE-2020-8276	4.9984	2.1	5.5
CVE-2020-9300	4.7012	4	6.5
CVE-2020-14040	5.5183	5	7.5
CVE-2020-16121	5.1676	2.1	3.3
CVE-2020-24990	5.1261	5	7.5
CVE-2020-17082	6.9564	7.5	9.8
CVE-2020-17086	6.9564	7.5	9.8
CVE-2020-17081	4.5548	5	7.5
CVE-2020-17079	6.9564	7.5	9.8
CVE-2020-17078	6.9564	7.5	9.8
CVE-2020-17083	6.9546	3.5	5.4
CVE-2020-17084	6.9546	9	8.8
CVE-2020-4568	3.1244	2.1	5.5
CVE-2020-0409	5.811	4.6	7.8
CVE-2020-17087	4.8916	7.2	7.8

Figure 6 Prediction vs. Actuals

VI. CONCLUSION AND FUTURE WORK

The current model can be used to estimate a possible score for the vulnerability as soon as it has been disclosed. This model can be further improved using other techniques and additional features. Here are some ideas for future work

- Use combination of words (n-grams) to predict score
- Use CPE (Common Platform Enumeration) if available
- Create separate models for CVSS V2 and V3 prediction
- Use mutual info regression function to score and select features
- Use Neural Network for predicting score