Cybersecurity Analytics: Final Project

Quoc H. Nguyen, Ankit Shah*
University of South Florida, Tampa, USA

Abstract

Index Terms

CVSS Prediciton, Cybersecurity Analytics

I. Project Description

A report on cyber network vulnerabilities includes 23,918 records. Each data sample represents a distinct occurrence of a vulnerability identified on a device in a computer network. This report includes details such as the name and description of the vulnerability occurrences, protocol types, port numbers, a synopsis, a potential solution, and website links to learn more. The Common Vulnerability Scoring System (CVSS) value for each vulnerability incidence is the response variable, ranging from 1 to 10. The criticality of a vulnerability instance discovered on a device is determined by this value, with 10 being the most critical and 1 being the least critical.

Objective: The goal of this project is to learn from historical data to estimate the CVSS values of future/unseen vulnerability cases in order to identify their respective criticality.

II. Methods

The study undertakes the task of CVSS prediction. The pipeline of the proposed approach is illustrated in Figure. 1. First, we perform Explotary Data Analysis (EDA), which refers to the critical process of discovering initial investigations such as finding patterns, anomalies, and summarize main characteristics of the dataset. Next, data pre-processing is implemented including removing stopwords, steaming, lemmatization and converting categorical data. Then we perform feature engineering to extract insights from the mixed data. After that, we build and examine different models to predict CVSS such as tree-based and non-linear models.

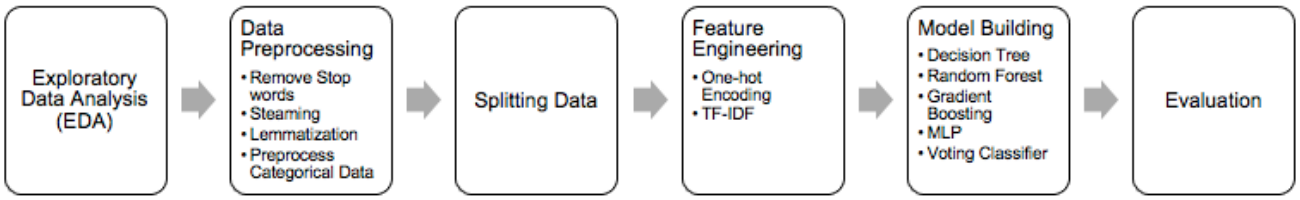


Fig. 1: The processing pipeline.

A. Exploratory Data Analysis

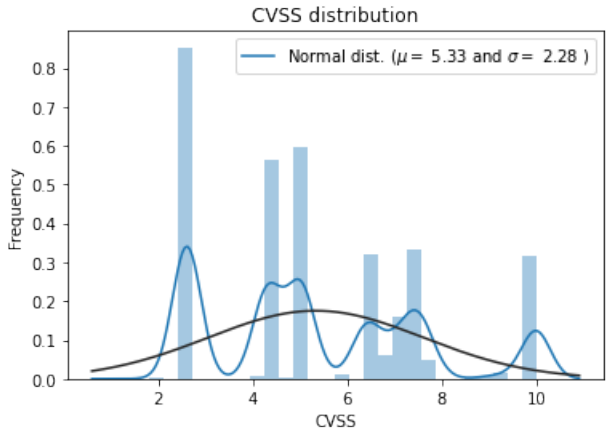
The dataset contains a mixed type of variables including categorical and text data. As shown in Fig. 2a, the CVSS distribution is a little bit right skew and not follow a normal distribution as data points dot not fit the straight line in QQ-plot. Linear-based models is not a good solution in this case, compared to other models such as tree-based models which does not required data normally distributed. The Plugin ID and CVSS scatter plot does not have any pattern, indicating that Plugin ID is not a significant feature to predict CVSS. In addition, Port numbers ranging from 0 to 10000 and 50000 to 60000 are likely have vulnerability events. We can consider this Port number as a categorical feature for building predicting models. CVE has some NaN values, to convert CVE to categorical, we would replace them to ”unknown” values.

B. Data Preprocessing

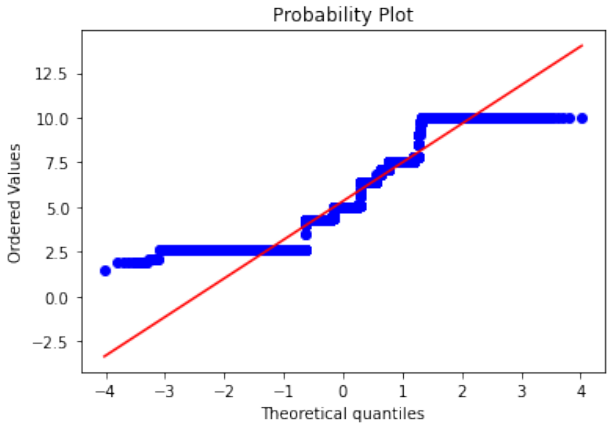
The dataset is not clean, containing noise and missing values. In this session, we employ the following techniques to pre-process the data.

1) Tokenization: Tokenization is a common technique that split a sentence into tokens, where a token could be characters, words, phrases, symbols, or other meaningful elements. By breaking sentences into smaller chunks, that would help to investigate the words in a sentence and also the subsequent steps in the NLP pipeline

Fig. 2: CVSS distribution and QQ-plot

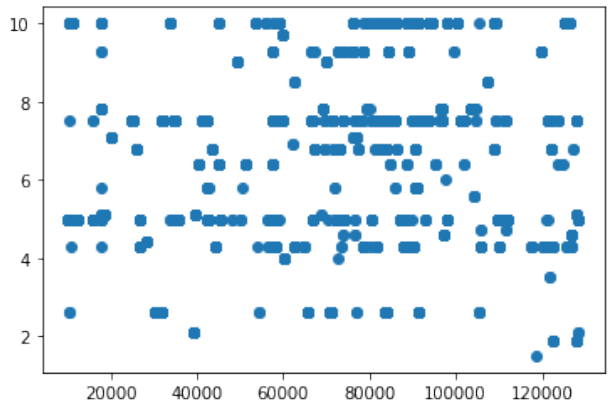


(a) CVSS distribution

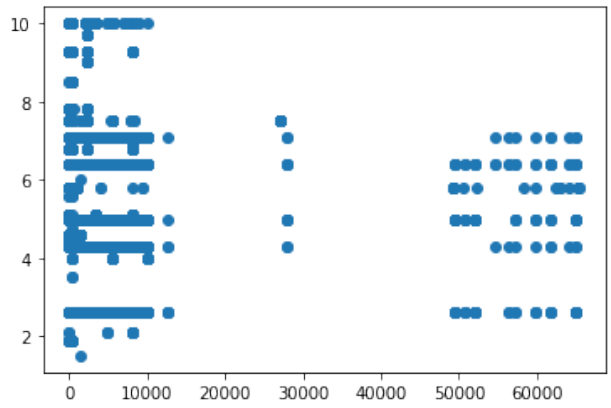


(b) QQ-plot of CVSS

Fig. 3: Scatter plot of Plugin ID and Port with the taget variable CVSS



(a) Plugin ID and CVSS



(b) Port and CVSS

2) Remove Stop Words: Stop words are common words in any language that occur with a high frequency but do not deliver meaningful information for the whole sentence. For example, “a”, “about”, “above”, “across”, “after”, “afterward”, “again”, ... can be considered as stop words. Traditionally, we could remove all of them in the text preprocessing stage. Removing stop words is a common method in NLP text preprocessing, whereas, it needs to be experimented carefully depending on different situations. However, removing stopwords is depended on a differnt senarios. For example, in this dataset, removing stopwords may cause losing critical information.

3) Steaming: Stemming is a process of extracting a root word - identifying a common stem among various forms (e.g., singular and plural noun form) of a word

4) Lemmatization: Lemmatization is the task of determining that two words have the same root, despite their surface differences

5) Preprocessing Categorical Columns: We fill missing ”CVE” values with ”unknown” values and convert categorical columns as ”category” type.

C. Splitting data

We divide the dataset into traning and testing with a spatio of 80% and 20%, repsectivly. In addition, the dataset was shuffled for better model performance.

D. Feature Engineering

1) On-hot Encoding: One hot encoding can be defined as the essential process of converting the categorical data variables to be provided to machine and deep learning algorithms which in turn improve predictions as well as classification accuracy of

a model. One Hot Encoding is a common way of preprocessing categorical features for machine learning models. We convert Plugin ID, CVE, Protocol and Port into numerical values using one-hot encoding.

2) TF-IDF: Tf-idf stands for term frequency-inverse document frequency, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the tf-idf weighting scheme are often used by search engines as a central tool in scoring and ranking a document's relevance given a user query. There are four text columns in the dataset including Name, Synopsis, Description and Solution. We used Tf-idf converter, a sklearn function to convert them into a dataframe features. To handle a large amount of features generated from one-hot encoding and tf-idf vecterizor, we use ColumnTransfomer, a function in python, to easily combine all features to feed into machine learning models. The feature engineering process is shown in the Figure. 4.

```
MAX_FEAT_DESCP = 50000
preprocess = ColumnTransformer(
    [
     ('cve_category', OneHotEncoder(dtype='int', handle_unknown='ignore'), ['CVE']),
     ('port_category', OneHotEncoder(dtype='int', handle_unknown='ignore'), ['Port']),
     ('plugin_id_category', OneHotEncoder(dtype='int', handle_unknown='ignore'), ['Plugin ID']),
     ('protocol_category', OneHotEncoder(dtype='int', handle_unknown='ignore'), ['Protocol']),
     ('name_tfidf', TfidfVectorizer(max_features = MAX_FEAT_DESCP, stop_words = None, ngram_range=(1,4)), 'Name'),
     ('synopsis_tfidf', TfidfVectorizer(max_features = MAX_FEAT_DESCP, stop_words = None, ngram_range=(1,4)), 'Synopsis'),
     ('solution_tfidf', TfidfVectorizer(max_features = MAX_FEAT_DESCP, stop_words = None, ngram_range=(1,4)), 'Solution'),
     ('des_tfidf', TfidfVectorizer(max_features = MAX_FEAT_DESCP, stop_words = None, ngram_range=(1,4)), 'Description')
    ],
    remainder='passthrough'
    )
```

Fig. 4: Combing categorical and text features using Column Transformer.

E. Model Building

In this session, we implemented different models such as Ridge regression, Decision Tree, Gradient Boosting and Multi-Layers Perceptron models. Then, we build an emsemble voting model to avergae learning from all models. The process is described in Figure. 5. The pro and cons are also described in the figure. To maximize the performance, we need to tune hyper-parameters using GridSearch for all model, and then retrain to input to Voting Classifer.

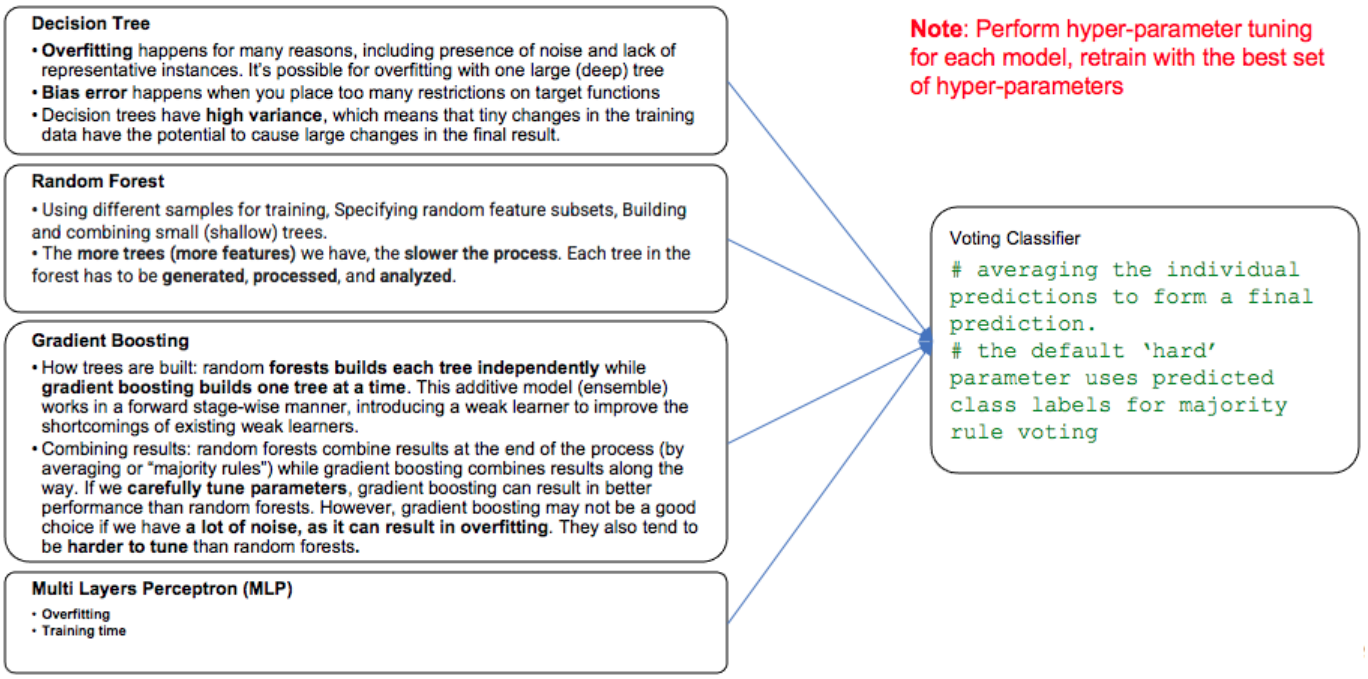


Fig. 5: Voting Classifier strategy

III. Results

We use RMSE as the metric to evaluate models. The predicting result on testing dataset is shown in Figure. 6. We can see that Decision Tree and Random Forest has similar RMSE score as 0.1174 and 0.1171, respectively, but decision tree is faster than random forest as it takes an average of many weaker trees. Gradient Boosting does not perform well with the RMSE score of 0.2596. The more complex model with MLP give us a prety good RMSE score but it takes a long time for training. Finally, with voting classifer, this ensemble method take the avergae of all predicting sample output from all models, giving us the best RMSE score as 0.196.

Models	RMSE	Training Time (s)
Decision Tree	0.1174	14.7989
Random Forest	0.1171	293.6796
Gradient Boosting	0.2596	85.5402
MLP	0.1207	400.4982
Voting Classifier	**0.1096**	563.2141

Fig. 6: RMSE score and training time of all examined models

IV. Future Works

The CVSS values is not balanced. This approach is for regression. We can convert the input variables into categorical problems, where we can apply sampling techniques such as SMOTE to make the dataset become more balanced. There are also other techniques for in-balance regression dataset such as SMOGN.

References