Cyber Threat Detection (Final Report)

Problem Statement

Using the Cyber Threat Dataset: Network, Text & Relation (via Kaggle - https://www.kaggle.com/datasets/ramoliyafenil/text-based-cyber-threat-detection) with a comprehensive collection of data for detecting, diagnosing, and mitigating cyber threats using network traffic data, textual content, and entity relationships. I will be training a machine learning model to identify and classify various types of cyber threats based on network traffic data and textual content.

Data Wrangling

The raw dataset contains 19940 rows and 8 columns, nearly all the columns aside from the text column had missing values.

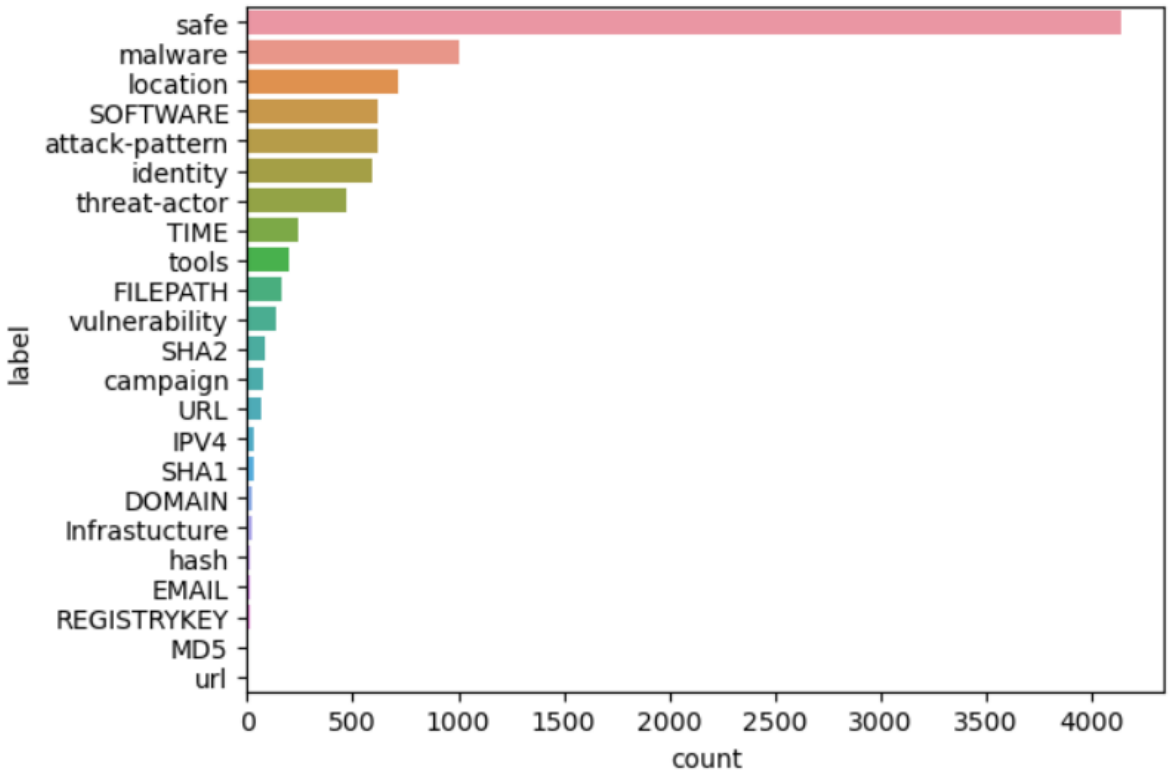
The various columns are:

- text – Textual content transferred over the network, such a emails, messages, or network traffic payloads. This column may contain description of potential cyber threats or attack vectors.
- relations – A list of tuples representing the relationships between entities, where each tuple contains a pair of entity IDs indicating the source and target of the relationship.
- Comments – A description or diagnosis of the identified cyber threat, providing insights into the nature and potential impact of the threat.
- entities – A list of JSON objects containing the below fields.
- id – The ID of the entity that sent or initiated the communication.
- label – The type of cyber threat or attack pattern identified, such as malware, attack pattern, identity, benign, software attack, or threat actor.
- start_offset – The starting character position of the identified entity or threat within the text field.
- end_offset – The ending character position of the identified entity or threat within the text field.

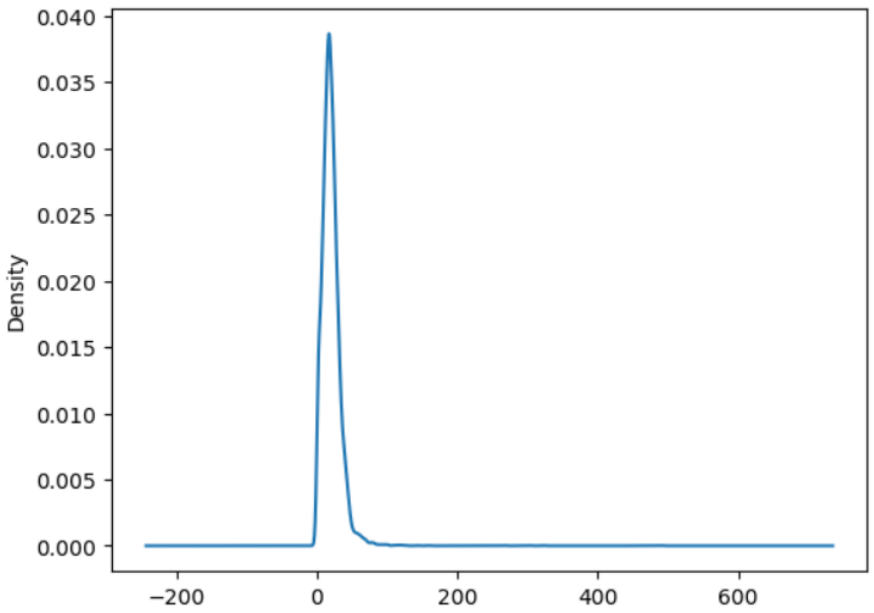
The missing values in the label column were filled with the string “safe” assuming all the cyber threats have been identified and the missing value rows do not contain any cyber threat information. The remaining missing values in the other columns were filled with an empty string as a place holder for N/A, no information available. Next, I checked for duplicate rows, there were 9269 duplicate rows that were dropped. The dataset is cleaned and now ready for further analysis, it was saved to a new CSV file ‘Wrangled.csv’.

Explanatory Data Analysis

For Natural Language Processing (NLP), I will focus on the text and label columns of the data and create a new DataFrame “activity”. I first checked all the unique values in the label column and counted the number of occurrences for each label.

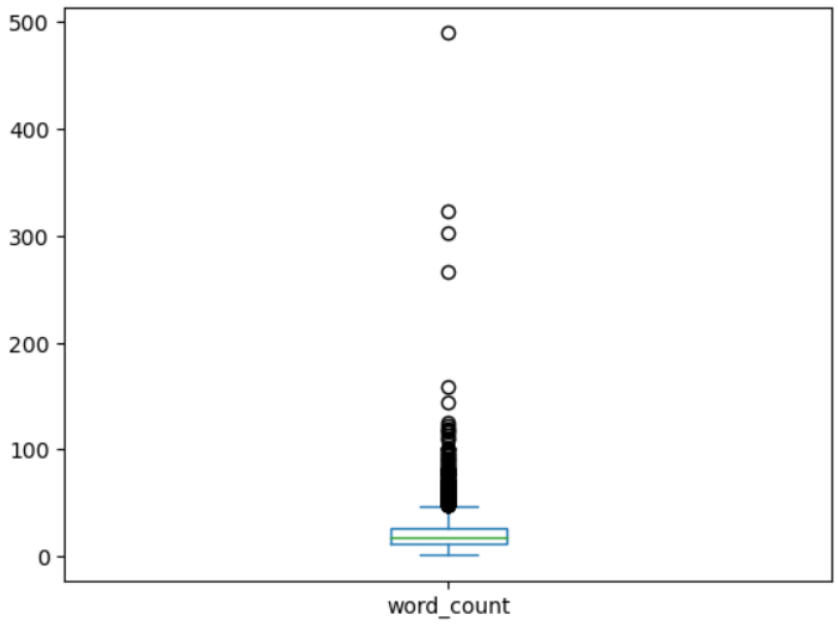


Based on the chart above, majority of the data is safe. There are around 1003 occurrences of malware, 617 occurrences of attack-patter, and 466 occurrences of threat-actor. You can see that the data is very imbalanced.



	word_count
count	9269.000000
mean	20.257633
std	14.459955
min	1.000000
25%	12.000000
50%	18.000000
75%	26.000000
max	490.000000

There is a total of 187768 words and on average the word count is around 20 words for each text. The label with the most words is ‘tools’ wiith max 490 words in a text. Majority of the text data falls between 1-100 words.



Further checking the word count data, there are a few outliers with very high word count to 490 words, some with range 275 to 350 words.

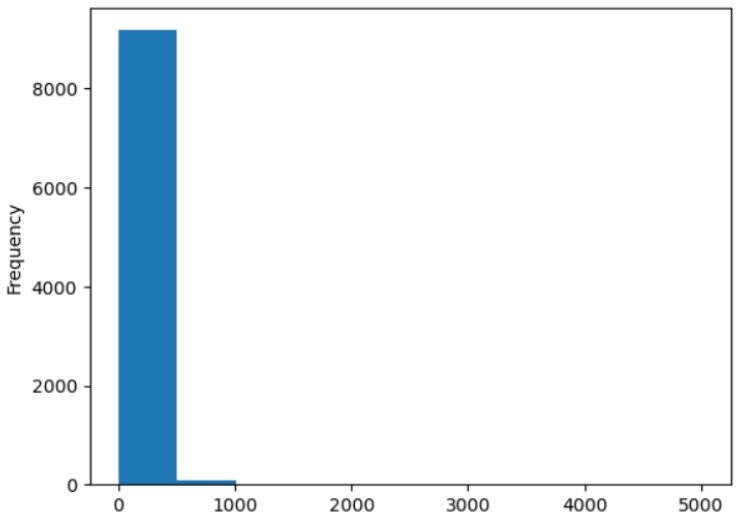
The top 5 rows with the longest word count:

	text	label	word_count
	Administrative Office of the President, Afghan...	tools	490
	Command Behavior Compress_Files Compresses...	FILEPATH	323
	systemsetting.exe BIOPASS RAT binary (PyInsta...	malware	303
	Indicators of Compromise Indicator Descripti...	threat-actor	266
	Several adversarial techniques were observed ...	identity	159

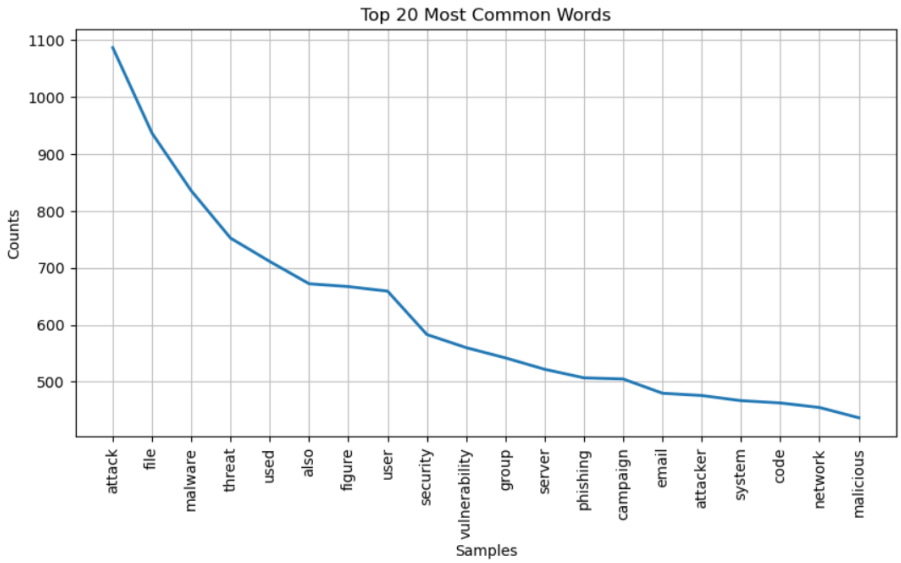
The top 5 rows with the shortest word count:

	text	label	word_count
5505	StealthWorker	safe	1
3366	flash.exe	malware	1
2360	ScrambleCross	malware	1
3377	Win64.BIOPASS.A	malware	1
7727	Buffer	safe	1

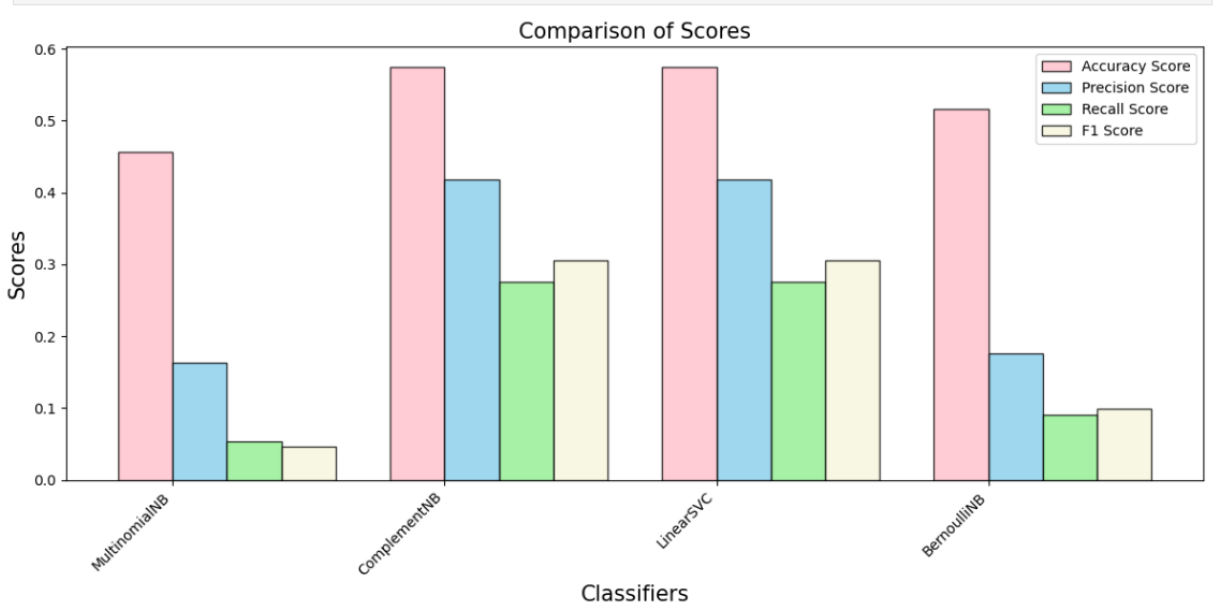
	label	word_count	char_length
16	location	27.693390	199.960619
22	vulnerability	26.814815	185.348148
21	url	26.750000	178.250000
6	REGISTRYKEY	26.400000	225.800000
19	threat-actor	25.444206	177.510730
3	IPV4	25.322581	233.387097
20	tools	24.631841	172.427861
15	identity	24.121417	158.956155
9	SOFTWARE	23.659711	151.707865
2	FILEPATH	23.648148	195.777778
13	campaign	23.211268	152.704225
4	Infrastucture	22.666667	147.291667
12	attack-pattern	22.544571	154.936791
10	TIME	22.481633	141.016327
17	malware	20.938185	149.745763
7	SHA1	20.900000	426.133333
14	hash	19.461538	258.153846
0	DOMAIN	19.400000	140.160000
8	SHA2	18.862500	352.412500
5	MD5	17.750000	276.375000
1	EMAIL	17.000000	145.166667
18	safe	16.167150	102.702415
11	URL	13.169231	180.046154



Checking the frequency of the character length, it shows similar results as the word length, majority of the data ranging between 1 to 500 characters long.

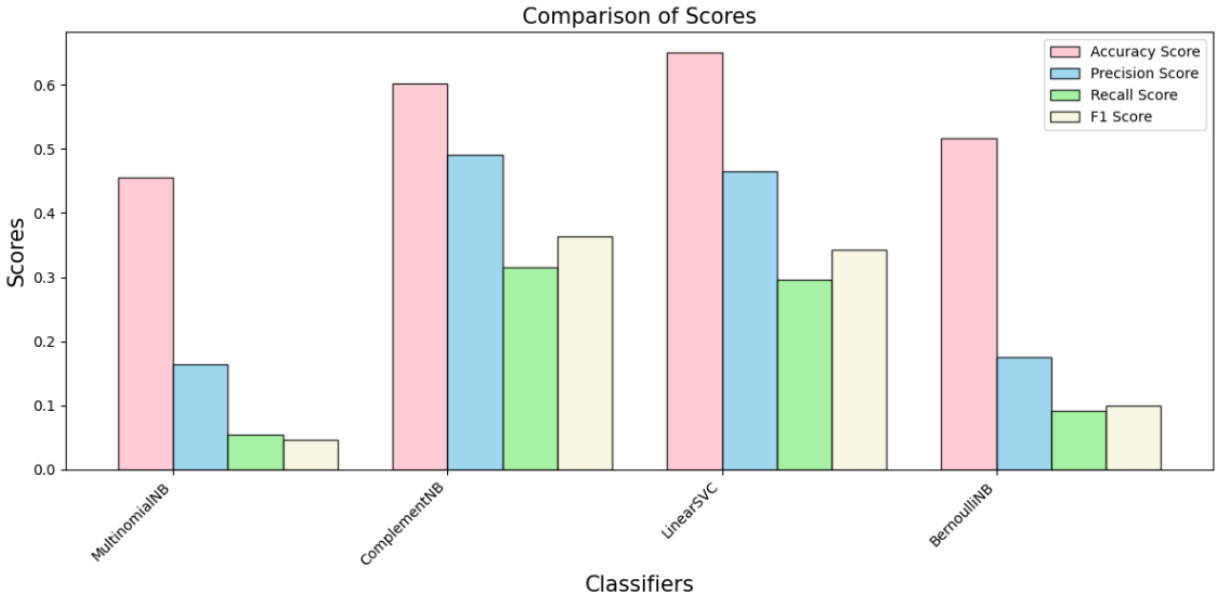


Pre-processing & Modeling



- MultinomialNB: Lowest accuract score, precision score, recall score, and F1-score.
- ComplementNB & LinearSVC: Identical accuracy score, precision score, recall score, and F1-score.
- BernoilliNB: Slightly lower accuracy score but comparable precision score, recall score, and F1-score to ComplementNB and LinearSVC.

Overall, ComplementNB and LinearCSV performed the best. I will next tune hyperparameters to further improve the performance of both models.



After hyperparameter tuning, it did increase the performance of both ComplementNB and LinearSVC.

- LinearSVC: Highest accuracy score and precision score overall.
- ComplementNB: Highest precision score but lower accuracy score than LinearSVC.
- BernoulliNB: Lowest scores overall.

LinearSVC seems to be the best performing model. We will test this model with new data to see if it can accurately predict labels for the given textual content.

```
new_data = "Maikspy on the Windows platform Figure 7."
#This is malware

preprocessed_data = tokenize_text(new_data)
print("Preprocessed data:", preprocessed_data)
```

```
Preprocessed data: maikspy window platform figure
```

```
predictions = best_model_SVC.predict([preprocessed_data])
predictions
```

```
array(['malware'], dtype=object)
```

```
new_data2 = "On the other hand, FIN7 performs hooking."
#This is attack-pattern

preprocessed_data2 = tokenize_text(new_data2)
print("Preprocessed data:", preprocessed_data2)
```

```
Preprocessed data: hand fin performs hooking
```

```
predictions2 = best_model_SVC.predict([preprocessed_data2])
predictions2
```

```
array(['attack-pattern'], dtype=object)
```

The model has successfully predicted the correct label for each text

Future Work

- Introduce additional cyber activity data and retrain the model to perform even better.
- Train the model to also produce recommended solutions along with the predicted labels.