RCE-Notebook-M15

July 13, 2022

0.0.1 RCE (Remote Code Execution) Attack Detection

RCE (Remote Code Execution) is an attacker's ability to run any commands or code of the attacker's choice on a target machine or in a target process. The machine learning model uses Python NLTK to build features and model. The use of Python NLTK will make the elaboration easier. In addition, scikit-learning module is also used to calculate some statistic calculation and comparison. It takes about 10 until 25 minutes to finish all jobs. In this research, there are two datasets.

1. https://github.com/payloadbox/command-injection-payload-list
2. https://www.kaggle.com/datasets/antonyj453/urldataset
3. https://github.com/swisskyrepo/PayloadsAllTheThings

Dataset References 1. Testing for Command Injection (OTG-INPVAL-013) 2. OWASP Command Injection 3. WE-77: Improper Neutralization of Special Elements used in a Command ('Command Injection') 4. WE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') 5. Portswigger Web Security - OS Command Injection

Importing Required Libraries

[ ]:
```
print('Installing required libraries...')
%pip install -q --upgrade pip
%pip install -q scikit-learn nltk wordcloud openpyxl python-slugify[unidecode]
print('Instalation done!')
```

```
Installing required libraries…
Note: you may need to restart the kernel to use updated packages.
Note: you may need to restart the kernel to use updated packages.
Instalation done!
```

[ ]:
```
import os
import csv
import json
import shutil
import pandas as panda
import numpy as np
import matplotlib.pyplot as plot
import nltk
from slugify import slugify
from nltk.tokenize import word_tokenize
```

```
# import sklearn functions
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC

from IPython.display import display, display_html, HTML
from tqdm.notebook import tqdm_notebook
from wordcloud import WordCloud
from datetime import datetime
```

[ ]:
```
dataset_payload     = os.path.join('datasets', 'Dataset-RCE.csv')
dataset_payload_dir = os.path.join('datasets', 'crawling')
dataset_nonpayload  = os.path.join('datasets', 'Dataset-Non-RCE.csv')
challenge_csv       = os.path.join('datasets', 'Dataset-RCE-Challenges.csv')
dataset             = os.path.join('datasets', 'Dataset-RCE-Non-RCE.csv')

report_dir          = os.path.join('reports')
reporter_dir        = os.path.join(report_dir, 'json')
classifier_dir      = os.path.join(report_dir, 'classifiers')
corpus_dir          = os.path.join('output', 'corpus')

if not os.path.exists(reporter_dir):
    os.makedirs(reporter_dir)
```

Preparing Dataset and Compilation

[ ]:
```
_dataset = [['Sentence', 'Label']]

def get_payload_file_content(payload_file, delimiter=None):
    contents = []
    with open(payload_file, 'r') as reader:
        data = reader.read()
        for line in data.split('\n'):
            if line:
                if delimiter:
                    line = line.split(delimiter)
                    if len(line) == 2:
                        [sentence, label] = line
                        if label.isdigit():
                            contents.append([sentence, label])
                else:
                    contents.append([line, '1'])
```

```
    return contents


# read payload command based file https://github.com/swisskyrepo/
 ↪PayloadsAllTheThings
print('Crawling payload...')
for payload_dir in os.listdir(dataset_payload_dir):
    payload_dir = os.path.join(dataset_payload_dir, payload_dir)
    for intruder_dir in os.listdir(payload_dir):
        if intruder_dir == 'Intruder':
            intruder_dir = os.path.join(payload_dir, intruder_dir)
            for payload_file in os.listdir(intruder_dir):
                payload_file = os.path.join(intruder_dir, payload_file)
                contents     = get_payload_file_content(payload_file)
                for content in contents:
                    _dataset.append(content)


print('Total of Payload Crawled:',  len(_dataset))
```

```
Crawling payload…
Total of Payload Crawled: 23582
```

[ ]:
```
print('Combining payload and non-payload dataset...')

# read payload dataset
with open(dataset_payload, 'r') as reader:
    csv_payload = csv.reader(reader)
    for line in csv_payload:
        if line: _dataset.append([line[0], "1"]) # adding payload label using␣
 ↪"1"


# get total dataset of payload so it can be 50:50 with non payload dataset
payload_dataset_total = len(_dataset)

# read non payload dataset
counter, counter_np = 0, 0
with open(dataset_nonpayload, 'r') as reader:
    csv_payload = csv.reader(reader)
    for line in csv_payload:
        if len(line) == 2:
            url, label = line
            if label == 'good':
                _dataset.append([url, "0"]) # adding non-payload label using "0"
                counter += 1
        if counter > payload_dataset_total:
```

```
            break

# write payload and non payload dataset into one file
with open(dataset, 'w') as writer:
    csv_rce = csv.writer(writer)
    for item in _dataset:
        csv_rce.writerow(item)


print('Total of Dataset Row\t\t:',    (len(_dataset) - 1))
print('Total of Payload Rows\t\t:',   len([sentence for sentence, label in␣
 ↪_dataset if label == '1']))
print('Total of Non Payload Rows\t:', len([sentence for sentence, label in␣
 ↪_dataset if label == '0']))


del _dataset # remote unused variable
```

```
Combining payload and non-payload dataset…
Total of Dataset Row            : 48192
Total of Payload Rows           : 24095
Total of Non Payload Rows       : 24097
```

Processing Mixed Database for Detection Model

[ ]:
```
df = panda.read_csv(dataset)
df.sample(n=5) # view dataset sample
```

[ ]:

	Sentence	Label
45041	collegecandy.com/2011/03/28/holy-hell-bikini-s…	0
22314	…/…/…/etc/passwd	1
9936	%2e%2e0x2f%2e%2e0x2f%2e%2e0x2f%2e%2e0x2fboot.ini	1
6758	%5C..%uF025%5C..%uF025etc%uF025passwd	1
43650	chroniclingamerica.loc.gov/lccn/sn86063624/189…	0

Cleaning and Positioning Dataset This step is for cleaning and positioning dataset, such as drop unused columns, remove empty values, and drop duplicate values.

[ ]:
```
# remove NAN or empty values in the dataset
data = df.dropna()
data.reset_index(drop=True, inplace=True) # reset index/counter
data.drop_duplicates(subset='Sentence', inplace=True) # drop duplicate values
data['Label'] = panda.to_numeric(data['Label']) # convert to numeric
data.sample(n=5)
```

[ ]:

	Sentence	Label
4496	.?%c1%af.?%c1%af.?%c1%af.?%c1%af.?%c1%…	1
8161	%2e.%u2216%2e.%u2216%2e.%u2216%2e.%u2216window…	1

```
7620                              .%2e%f0%80%80%afboot.ini      1
4551                          .?%u2216.?%u2216etc%u2216issue    1
37867  books.google.com/books/about/Whipping_girl.htm…      0
```

[ ]:
```python
# get more info about dataset which has been clean
rd_count = df[df.columns[0]].count() # real dataset
cd_count = data[data.columns[0]].count() # cleaned dataset

print("Total of original rows\t:", rd_count)
print("Total of eligible rows\t:", cd_count)
print("Total of removed rows\t:", rd_count - cd_count)
```

```
Total of original rows  : 48192
Total of eligible rows  : 47177
Total of removed rows   : 1015
```

[ ]:
```python
# remove previous dataset for avoid error
if os.path.exists(corpus_dir):
    shutil.rmtree(corpus_dir)

pbar  = tqdm_notebook(total=len(data.values), desc="Convert CSV to corpus file")
index = 1
for payload, label in data.values:
    label     = 'payload' if label == 1 else 'non-payload'
    label_dir = os.path.join(corpus_dir, label)
    if not os.path.exists(label_dir):
        os.makedirs(label_dir)
    dt_filepath = os.path.join(label_dir, f'{index}.txt')
    with open(dt_filepath, 'w') as writer:
        writer.write(payload)
    pbar.update(1)
    index += 1
pbar.close()


del df, data
```

```
Convert CSV to corpus file:   0%|          | 0/47177 [00:00<?, ?it/s]
```

Core Steps of Model or Classifier Building

[ ]:
```python
from nltk.corpus.reader import CategorizedPlaintextCorpusReader

corpus_set = CategorizedPlaintextCorpusReader(
    corpus_dir, r'(?!\.).*\.txt',
    cat_pattern=r'(payload|non-payload)/.*',
    encoding="utf-8"
)
```

```
print('Payload Total\t\t:',   len(corpus_set.fileids('payload')))
print('Non-Payload Total\t:', len(corpus_set.fileids('non-payload')))
print('Total Corpus\t\t:',    len(corpus_set.fileids('payload')) +␣
 ↪len(corpus_set.fileids('non-payload')))
```

```
Payload Total           : 23080
Non-Payload Total       : 24097
Total Corpus            : 47177
```

Corpus Cleaning and Positioning

[ ]:
```
import string
import re, inspect

corpus_rules = {
    'to_lowercase'  : True,
    'only_alphanum' : False,
    'remove_puncts' : False,
}


def construct_corpus(corpus_in)->list:
    punctuations = list(string.punctuation)
    corpus_out   = []
    if type(corpus_in) == str:
        if corpus_rules['to_lowercase']:
            corpus_in = corpus_in.lower()
        corpus_in = word_tokenize(corpus_in)
    else:
        if corpus_rules['to_lowercase']:
            corpus_in = [w.lower() for w in corpus_in]
    for item in corpus_in:
        if not corpus_rules['remove_puncts']:
            punctuations = []
        if not item in punctuations:
            if corpus_rules['only_alphanum']:
                item = re.sub(r'[\W\d]', '', item)
            if item:
                corpus_out.append(item)
    return corpus_out


def construct_features(corpus:string)->dict:
    feature = {}
    corpus  = set(construct_corpus(corpus))
    for vector in feature_vectors.keys():
        feature[vector] = vector in corpus # corpus string -> corpus list
```

```
    return feature


def check_payload(classifier, payload:string):
    predict = classifier.classify(construct_features(payload))
    return predict


def get_accuracy(classifier, test_fs):
    predict = classifier.classify_many([fs for (fs, l) in test_fs])
    correct = [l == r for ((fs, l), r) in zip(test_fs, predict)]
    if correct:
        accuracy = sum(correct) / len(correct)
    else:
        accuracy = 0
    return [predict, accuracy]


def wordcloud_draw(data, filename, bgcolor='white'):
    words  = ' '.join(data)
    wCloud = WordCloud(background_color=bgcolor, width=2250, height=1000).
 ↪generate(words)
    wCloud.to_file(filename)
    plot.figure(1, figsize=(20, 10))
    plot.imshow(wCloud)
    plot.axis('off')
    plot.show()


def wordplot_draw(freq_words, filename, graph_limit=30):
    fig = plot.figure(figsize=(16, 7))
    plot.gcf().subplots_adjust(bottom=0.15)
    freq_words.plot(graph_limit, title=f'{graph_limit} Most Common Words')
    plot.show()
    plot.draw()
    fig.savefig(filename, dpi=120)


def save_report(classifiers, report_name):
    filename = os.path.join(reporter_dir, f"{report_name}.json")
    reports  = []
    for id, attr in classifiers.items():
        item = {k:str(v) for k,v in attr.items()}
        reports.append(item)
    if not reports:
        return None
    with open(filename, 'w') as writer:
```

```
        json.dump(reports, writer)


def get_report_history():
    reports = []
    for report_file in os.listdir(reporter_dir):
        filename = os.path.join(reporter_dir, report_file)
        if filename.endswith('.json'):
            with open(filename) as reader:
                report   = json.load(reader)
                reports += report
    return reports


def show_report_history(n=10):
    reports = get_report_history()
    if reports:
        tf = panda.DataFrame.from_records([item.values() for item in reports],␣
 ↪columns=[list(reports[0].keys())])
        filepath  = os.path.join(os.getcwd(), report_dir, 'report')
        tf.to_excel(f"{filepath}.xlsx")
        tf.to_html(f"{filepath}.html")
        display(tf.head(n))
```

[ ]:
```
pbar          = tqdm_notebook(corpus_set.fileids(), desc="Calculating words␣
 ↪count")
words_counter = []
for file_id in corpus_set.fileids():
    word_count = len(corpus_set.words(fileids=file_id))
    words_counter.append(word_count)
    pbar.update(1)
pbar.close()

margin       = 15
vector_limit = int((sum(words_counter)/len(words_counter)) * margin)
print(f'Vector limit is {vector_limit}')
```

```
Calculating words count:   0%|          | 0/47177 [00:00<?, ?it/s]

Vector limit is 365
```

[ ]:
```
feature_vectors = {}
_freq_words     = {} # for informational purpose

for category in corpus_set.categories():
    print(f"Injecting {vector_limit} {category} vectors to feature vectors...")
```

```
    corpus_cat = corpus_set.words(categories=[category])
    freq_words = nltk.FreqDist(construct_corpus(corpus_cat))
    most_words = {word.lower():n for word, n in freq_words.
 ↪most_common(vector_limit)}
    _freq_words[category] = [most_words, freq_words]
    feature_vectors.update(most_words)

    t = {f"{word.lower()},{n}" for word, n in freq_words.most_common(50)}
    display(t)


# save feature fectors
filename = f"feature_vector-tfs({len(corpus_set.
 ↪fileids())})-vl({vector_limit})-m({margin})"
cr_name  = '-'.join([f"{k}({int(v)})" for k,v in corpus_rules.items()])
with open(os.path.join(classifier_dir, f'{filename}-{cr_name}.json'), "w") as␣
 ↪writer:
    json.dump(feature_vectors, writer)
    print('Feature vector saved...')
```

```
Injecting 365 non-payload vectors to feature vectors…
```

```
{'%,1778',
 '&,2101',
 '+,1640',
 '-,43562',
 '.,49349',
 '/,56071',
 '/?,465',
 '08,445',
 '09,443',
 '1,493',
 '10,642',
 '11,583',
 '2009,520',
 '2010,854',
 '2011,2011',
 '=,5642',
 '?,3128',
 'a,447',
 'about,497',
 'ancestry,604',
 'and,500',
 'answers,1071',
 'articles,1194',
 'aspx,615',
 'blog,903',
 'blogs,701',
```

```
 'blogspot,1533',
 'ca,2064',
 'com,19399',
 'content,511',
 'edu,668',
 'htm,994',
 'html,3837',
 'id,541',
 'in,1000',
 'index,1579',
 'linkedin,1010',
 'net,681',
 'news,518',
 'of,605',
 'org,2069',
 'p,442',
 'php,1043',
 'pub,700',
 'qid,1022',
 'question,1036',
 'tag,568',
 'the,933',
 'wiki,565',
 'yahoo,1359'}

Injecting 365 payload vectors to feature vectors…

{'%%,24883',
 '%,323227',
 '.%,13368',
 '.,6294',
 '..%,7417',
 '…/%,2016',
 '…/.%,2016',
 '.?%,2016',
 '/,2379',
 '00,2591',
 '01,2352',
 '252e,5448',
 '252f,1764',
 '255c,1764',
 '25ae,5448',
 '25af,1764',
 '25c0,9120',
 '25c1,3504',
 '2e,16543',
 '2f,3805',
 '32,9118',
 '35,3551',
```

```
 '5c,6311',
 '5e,4872',
 '63,1829',
 '65,5459',
 '66,1794',
 '6e,4900',
 '80,70560',
 '?.%,2016',
 '??%,2016',
 '\\,2082',
 'ae,25104',
 'af,8262',
 'afetc,2520',
 'bg,3360',
 'c0,53850',
 'c1,16944',
 'e0,8400',
 'ee,4872',
 'etc,1784',
 'f0,8400',
 'f8,8400',
 'fc,5040',
 'fe,4872',
 'ini,5313',
 'qf,3240',
 'u2215,1764',
 'u2216,1764',
 'uff0e,5448'}
```

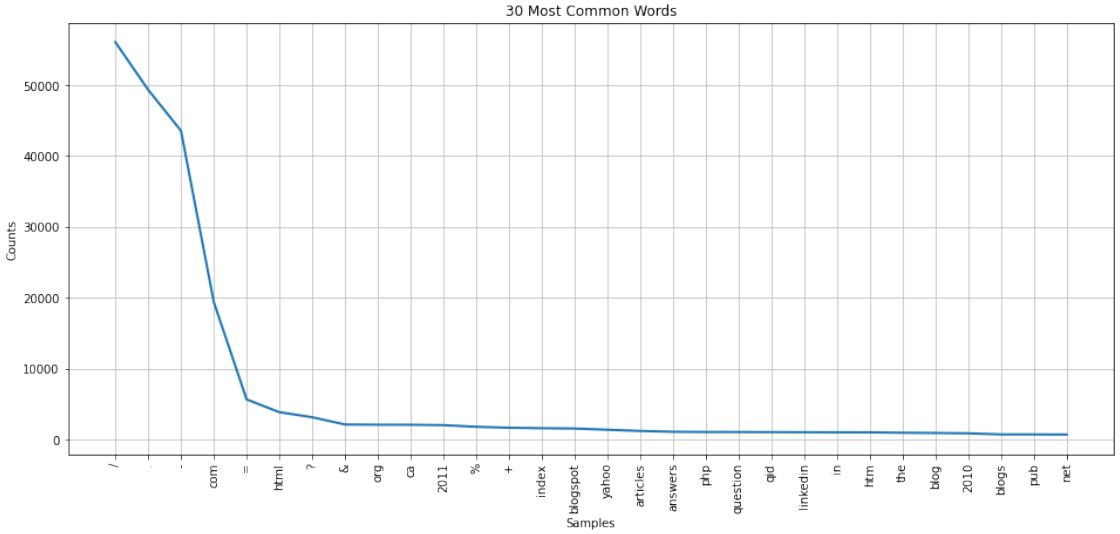
Feature vector saved…

Informational Purpose

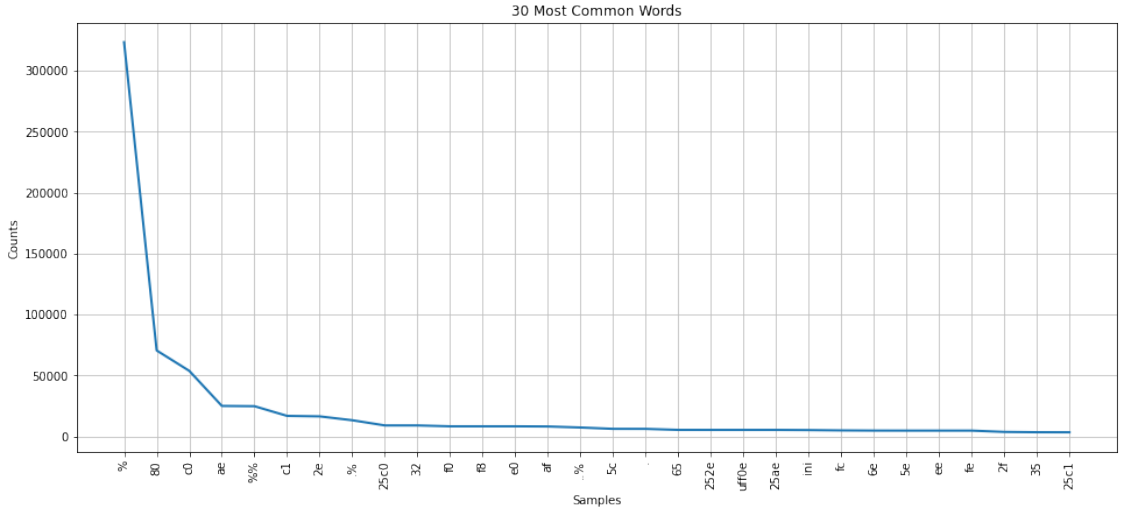
[ ]:
```
graph_limit = 30
figure_dir  = os.path.join('reports', 'figures')
if not os.path.exists(figure_dir):
    os.makedirs(figure_dir)

for category, attrs in _freq_words.items():
    print(f'Showing {vector_limit} most common words in {category}...')
    most_word, freq_words = attrs
    filename = f"{category}-tfs({len(corpus_set.
 ↪fileids())})-vl({vector_limit})-m({margin})"
    cr_name  = '-'.join([f"{k}({int(v)})" for k,v in corpus_rules.items()])
    wordcloud_draw(most_word, os.path.join(figure_dir,␣
 ↪f'WC-{filename}-{cr_name}.png'))
    wordplot_draw(freq_words, os.path.join(figure_dir,␣
 ↪f'WP-{filename}-{cr_name}.png'), graph_limit)
```

```
del _freq_words
```

Showing 365 most common words in non-payload…



Showing 365 most common words in payload…



<Figure size 432x288 with 0 Axes>

Preparing Feature Enginering Functions

[ ]:
```
feature_set_dir = os.path.join('output', 'feature-sets')

# remove previous feature_set_dir for avoid error
if os.path.exists(feature_set_dir):
    print('Removing old feature-sets...')
    shutil.rmtree(feature_set_dir)

if not os.path.exists(feature_set_dir):
    os.makedirs(feature_set_dir)
```

```
datasets  = []
pbar      = tqdm_notebook(corpus_set.fileids())
step      = 1
for category in corpus_set.categories():
    pbar.set_description(f"({step}/{len(corpus_set.categories())}) Classifying␣
 ↪Courpus -> {category}")
    labels = corpus_set.fileids(category)
    for file_id in labels:
        content = corpus_set.raw(file_id)
        datasets.append((file_id, content, category))
        pbar.update(1)
    step += 1
pbar.close()
```

```
Removing old feature-sets…

  0%|          | 0/47177 [00:00<?, ?it/s]
```

[ ]:
```
feature_sets = []
pbar         = tqdm_notebook(total=len(datasets), desc=f"Processing Feature Set␣
 ↪with {vector_limit} Vector Limit")
for (file_id, payloads, category) in datasets:
    basename = os.path.basename(file_id)
    filename = os.path.join(feature_set_dir, basename)
    features = construct_features(payloads)
    # with open(filename, 'w') as writer: json.dump([features, category],␣
 ↪writer)
    feature_sets.append([features, category])
    pbar.update(1)
pbar.close()

del datasets
```

```
Processing Feature Set with 365 Vector Limit:   0%|          | 0/47177 [00:00<?,␣
 ↪?it/s]
```

Spliting Data Training and Testing

[ ]:
```
from sklearn import model_selection


test_size = 0.2 # percentation of data testing
X, y      = [x for x,y in feature_sets], [y for x,y in feature_sets]

# split data training and data testing using Python NLTK
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y,␣
 ↪test_size=test_size)
```

```
# melihat jumlah data training dan testing
total_train_set     = len(X_train)
total_test_set      = len(X_test)
total_feature_sets  = len(feature_sets)

print('Data Training\t:', total_train_set)
print('Data Testing \t:', total_test_set)

del feature_sets
```

```
Data Training   : 37741
Data Testing    : 9436
```

Training and Testing Data

[ ]:
```
from nltk import classify

scikitNLTK          = classify.scikitlearn.SklearnClassifier
classifiers         = {
    'naive-bayes'   : {
        'name'      : 'Naive Bayes',
        'trainer'   : classify.NaiveBayesClassifier.train,
    },
    'logistic-regress': {
        'name'      : 'Logistic Regression',
        'trainer'   : scikitNLTK(LogisticRegression()).train,
    },
    'gradient-boost': {
        'name'      : 'Gradient Boosting',
        'trainer'   : scikitNLTK(GradientBoostingClassifier()).train,
    },
    'sv-machine': {
        'name'      : 'Support Vector Machine',
        'trainer'   : scikitNLTK(SVC()).train,
    },
    'kn-neighbors'  : {
        'name'      : 'K-Nearest Neigbors',
        'trainer'   : scikitNLTK(KNeighborsClassifier(n_neighbors=3)).train,
    },
}
```

[ ]:
```
print(f"Training information -> vector_limit {vector_limit} | margin: {margin}␣
 ↪| test_size: {test_size}...")
train_featureset    = [[X_train[i], y_train[i]] for i in range(total_train_set)]
test_featureset     = [[X_test[i], y_test[i]] for i in range(total_test_set)]

counter = 1
```

```
for id, attr in classifiers.items():
    print(f"[{counter}] Training data using {attr['name']}...")
    start_time          = datetime.now()
    attr['classifier']  = attr['trainer'](train_featureset)
    attr['y_predict'], attr['accuracy'] = get_accuracy(attr['classifier'],␣
 ↪test_featureset)
    attr['elapse_time'] = str(datetime.now() - start_time)
    print(f"Accuracy Level\t: {attr['accuracy']}")
    print(f"Time Ellapsed\t: {attr['elapse_time']}\n")
    counter += 1


print(f'Ordering classifier based on accuracy level...')
index       = 1
classifiers  = {k: v for k, v in sorted(classifiers.items(), key=lambda item:␣
 ↪item[1]['accuracy'], reverse=True)}
for id, attr in classifiers.items():
    print(f"[{index}] {attr['name']} (Accuracy: {attr['accuracy']}) (Time␣
 ↪Ellapse: {attr['elapse_time']})")
    index += 1


del train_featureset, test_featureset
```

```
Training information -> vector_limit 365 | margin: 15 | test_size: 0.2…
[1] Training data using Naive Bayes…
Accuracy Level  : 0.9760491733785502
Time Ellapsed   : 0:00:24.265259

[2] Training data using Logistic Regression…
Accuracy Level  : 0.9862229758372192
Time Ellapsed   : 0:00:23.469021

[3] Training data using Gradient Boosting…
Accuracy Level  : 0.9830436625688851
Time Ellapsed   : 0:00:56.636608

[4] Training data using Support Vector Machine…
Accuracy Level  : 0.9880245866892751
Time Ellapsed   : 0:01:17.004226

[5] Training data using K-Nearest Neigbors…
```

Classifier Testing with Small Sampling

[ ]:
```
custom_payloads = [
    ("10000daysatributetotool.bandzoogle.com/fr_home.cfm", 0),
    ("cyber.law.harvard.edu/icann/la/archive/attendee.html", 0),
```

```
    ("custom.gps-data-team.com/canada/winery_(vignoble)_quebec/index.php?
 ↪poi_data=winery_(vignoble)_quebec&file=vignoble_quebec", 0),
    ("&lt;!--#exec%20cmd=&quot;/usr/bin/id;--&gt;", 1),
    (";system('cat%20/etc/passwd')", 1),
    ("%0a ping -i 30 127.0.0.1 %0a", 1),
]
```

[ ]:
```
for id, attr in classifiers.items():
    table_row, tracks = [], []
    for payload, label in custom_payloads:
        label      = 'payload' if label == 1 else 'non-payload'
        prediction = check_payload(attr['classifier'], payload)
        status     = label == prediction
        table_row.append([payload, label, prediction, status])
        tracks.append(status)
    table_frame = panda.DataFrame.from_records(table_row, columns=['Payload',␣
 ↪'Actual', 'Predict', 'Status'])
    valid_preds = []
    for status in tracks:
        if status is True: valid_preds.append(status)
    display_html(f"<h3>{attr['name']} Prediction Result ({len(valid_preds)}/
 ↪{len(tracks)})</h3><hr/>", raw=True)
    display(table_frame)
```

```
                                              Payload       Actual  \
0  10000daysatributetotool.bandzoogle.com/fr_home…  non-payload
1  cyber.law.harvard.edu/icann/la/archive/attende…  non-payload
2  custom.gps-data-team.com/canada/winery_(vignob…  non-payload
3        &lt;!--#exec%20cmd=&quot;/usr/bin/id;--&gt;      payload
4                       ;system('cat%20/etc/passwd')      payload
5                       %0a ping -i 30 127.0.0.1 %0a      payload

       Predict  Status
0  non-payload    True
1  non-payload    True
2  non-payload    True
3      payload    True
4      payload    True
5      payload    True
```

```
                                              Payload       Actual  \
0  10000daysatributetotool.bandzoogle.com/fr_home…  non-payload
1  cyber.law.harvard.edu/icann/la/archive/attende…  non-payload
2  custom.gps-data-team.com/canada/winery_(vignob…  non-payload
3        &lt;!--#exec%20cmd=&quot;/usr/bin/id;--&gt;      payload
4                       ;system('cat%20/etc/passwd')      payload
5                       %0a ping -i 30 127.0.0.1 %0a      payload
```

```
       Predict  Status
0  non-payload    True
1  non-payload    True
2  non-payload    True
3      payload    True
4      payload    True
5  non-payload   False

                                                Payload       Actual  \
0  10000daysatributetotool.bandzoogle.com/fr_home…  non-payload
1  cyber.law.harvard.edu/icann/la/archive/attende…  non-payload
2  custom.gps-data-team.com/canada/winery_(vignob…  non-payload
3        &lt;!--#exec%20cmd=&quot;/usr/bin/id;--&gt;      payload
4                        ;system('cat%20/etc/passwd')      payload
5                        %0a ping -i 30 127.0.0.1 %0a      payload

       Predict  Status
0  non-payload    True
1  non-payload    True
2  non-payload    True
3      payload    True
4      payload    True
5      payload    True

                                                Payload       Actual  \
0  10000daysatributetotool.bandzoogle.com/fr_home…  non-payload
1  cyber.law.harvard.edu/icann/la/archive/attende…  non-payload
2  custom.gps-data-team.com/canada/winery_(vignob…  non-payload
3        &lt;!--#exec%20cmd=&quot;/usr/bin/id;--&gt;      payload
4                        ;system('cat%20/etc/passwd')      payload
5                        %0a ping -i 30 127.0.0.1 %0a      payload

       Predict  Status
0  non-payload    True
1  non-payload    True
2  non-payload    True
3      payload    True
4      payload    True
5      payload    True

                                                Payload       Actual  \
0  10000daysatributetotool.bandzoogle.com/fr_home…  non-payload
1  cyber.law.harvard.edu/icann/la/archive/attende…  non-payload
2  custom.gps-data-team.com/canada/winery_(vignob…  non-payload
3        &lt;!--#exec%20cmd=&quot;/usr/bin/id;--&gt;      payload
4                        ;system('cat%20/etc/passwd')      payload
5                        %0a ping -i 30 127.0.0.1 %0a      payload

       Predict  Status
```

```
0  non-payload    True
1  non-payload    True
2  non-payload    True
3  non-payload   False
4      payload    True
5      payload    True
```

Classifier Challenges and Evaluation with Big Sampling

[ ]:
```
limit_per_label = int(total_train_set)/4 # limit challenge with min based on␣
 ↪data training total
_challenge      = [['Sentence', 'Label']]
_challenge_csv  = os.path.join('datasets', 'Dataset-RCE-Challenges-Raw.csv')
with open(_challenge_csv, 'r') as reader:
    csv_challenge = csv.reader(reader)
    for line in csv_challenge:
        _challenge.append([line, '1'])
        if len(_challenge) > (limit_per_label - 1):
            break

# writing into file
with open(challenge_csv, 'w') as writer:
    csv_challenge = csv.writer(writer)
    for line in _challenge:
        csv_challenge.writerow(line)

df_challenge = panda.read_csv(challenge_csv, encoding='utf-8')
df_challenge.sample(n=5)

del _challenge
```

[ ]:
```
df_challenge.drop_duplicates(inplace=True) # drop duplicate
df_challenge.dropna(subset='Sentence', inplace=True) # drop empty values
df_challenge['Label'] = panda.to_numeric(df_challenge['Label'])
display_html(f"<h3>Total data will be predicted by classifier:␣
 ↪{len(df_challenge)}</h3><hr/>", raw=True)
```

[ ]:
```
step = 1
pbar = tqdm_notebook()
for id, attr in classifiers.items():
    pbar.reset(total=len(df_challenge.values))
    pbar.set_description_str(f"({step}/{len(classifiers)}) Predicting with␣
 ↪{attr['name']}")
    tracks     = []
    start_time = datetime.now()
    for payload, label in df_challenge.values:
        label      = 'payload' if label == 1 else 'non-payload'
        prediction = check_payload(attr['classifier'], payload)
```

```
        status     = label == prediction
        tracks.append(status)
        pbar.update(1)
    valid_preds = []
    for status in tracks:
        if status is True: valid_preds.append(status)

    ellapse_time        = str(datetime.now() - start_time)
    attr['challenges']  = {
        'total_data'    : len(tracks),
        'total_valid'   : len(valid_preds),
        'total_invalid' : (len(tracks) - len(valid_preds)),
        'ellapse_time'  : ellapse_time,
    }
    percentation = int((len(valid_preds)/len(tracks)) * 100)
    information  = ' | '.join([f"{k.capitalize()}: {v}" for k,v in␣
 ↪attr['challenges'].items()])
    display_html(f"<div>{attr['name']} Prediction ({percentation}%)</div>",␣
 ↪raw=True)
    display_html(f"<div>{information}<hr/></div>", raw=True)
    step += 1
pbar.close()

total_payload_challenge = len(df_challenge)
del df_challenge
```

```
0it [00:00, ?it/s]
```

[ ]:
```
def get_confussion_matrix(TP, FP, FN):
    confussion_matrix = {}
    for label in sorted(labels):
        precision, recall = 0, 0
        if TP[label] == 0:
            f_measure = 0
        else:
            precision = float(TP[label]) / (TP[label] + FP[label])
            recall    = float(TP[label]) / (TP[label] + FN[label])
            f_measure = float(2) * (precision * recall) / (precision + recall)
        confussion_matrix[label] = {
            'label'     : label,
            'precision' : precision,
            'recall'    : recall,
            'f_measure' : f_measure,
        }
    return confussion_matrix
```

```
def table_confussion_matrix(confussion_matrix):
    table_rows         = [v.values() for k,v in confussion_matrix.items()]
    table_frame        = panda.DataFrame.from_records(table_rows,␣
 ↪columns=['Label', 'Precision', 'Recall', 'F-Measure'])
    return table_frame
```

[ ]:
```
from collections import Counter

step = 1
for id, attr in classifiers.items():
    nltk_cm = nltk.ConfusionMatrix(y_test, attr['y_predict'], True)
    labels  = {'payload', 'non-payload'}

    TP, FN, FP = Counter(), Counter(), Counter()
    for i in labels:
        for j in labels:
            if i == j:
                TP[i] += int(nltk_cm[i,j])
            else:
                FN[i] += int(nltk_cm[i,j])
                FP[j] += int(nltk_cm[i,j])

    confussion_matrix         = get_confussion_matrix(TP, FP, FN)
    attr['confussion_matrix'] = confussion_matrix
    print(nltk_cm.pretty_format())
    display(table_confussion_matrix(confussion_matrix))
    print(f"With accuracy level -> {attr['accuracy']}")
    display_html('<hr/>', raw=True)
    step += 1

    # remove unused variable
    del attr['y_predict']
```

```
             |    n      |
             |    o      |
             |    n      |
             |    -      |
             |    p    p |
             |    a    a |
             |    y    y |
             |    l    l |
             |    o    o |
             |    a    a |
             |    d    d |
-------------+-----------+
non-payload |<4820>  35 |
    payload |   83<4498>|
```

```
------------+-----------+
(row = reference; col = test)
```

	Label	Precision	Recall	F-Measure
0	non-payload	0.983072	0.992791	0.987907
1	payload	0.992279	0.981882	0.987053

With accuracy level -> 0.9874947011445527

```
            |    n      |
            |    o      |
            |    n      |
            |    -      |
            |    p    p |
            |    a    a |
            |    y    y |
            |    l    l |
            |    o    o |
            |    a    a |
            |    d    d |
------------+-----------+
non-payload |<4854>   1 |
    payload |  137<4444>|
------------+-----------+
(row = reference; col = test)
```

	Label	Precision	Recall	F-Measure
0	non-payload	0.972551	0.999794	0.985984
1	payload	0.999775	0.970094	0.984711

With accuracy level -> 0.9853751589656634

```
            |    n      |
            |    o      |
            |    n      |
            |    -      |
            |    p    p |
            |    a    a |
            |    y    y |
            |    l    l |
            |    o    o |
            |    a    a |
            |    d    d |
------------+-----------+
non-payload |<4810>  45 |
    payload |  104<4477>|
------------+-----------+
(row = reference; col = test)
```

```
         Label  Precision    Recall  F-Measure
0  non-payload   0.978836  0.990731   0.984748
1      payload   0.990049  0.977298   0.983632
```

```
With accuracy level -> 0.9842094107672743
```

```
            |    n      |
            |    o      |
            |    n      |
            |    -      |
            |    p    p |
            |    a    a |
            |    y    y |
            |    l    l |
            |    o    o |
            |    a    a |
            |    d    d |
------------+-----------+
non-payload |<4808>  47 |
    payload |  136<4445>|
------------+-----------+
(row = reference; col = test)
```

```
         Label  Precision    Recall  F-Measure
0  non-payload   0.972492  0.990319   0.981325
1      payload   0.989537  0.970312   0.979830
```

```
With accuracy level -> 0.9806061890631623
```

```
            |    n      |
            |    o      |
            |    n      |
            |    -      |
            |    p    p |
            |    a    a |
            |    y    y |
            |    l    l |
            |    o    o |
            |    a    a |
            |    d    d |
------------+-----------+
non-payload |<4854>   1 |
    payload |  240<4341>|
------------+-----------+
(row = reference; col = test)
```

```
         Label  Precision    Recall  F-Measure
0  non-payload   0.952886  0.999794   0.975776
1      payload   0.999770  0.947610   0.972991
```

```
With accuracy level -> 0.9744595167443832
```

[ ]:
```
print('Saving Accuracy Level into file...')

for id, attr in classifiers.items():
    attr['datetime']         = datetime.now()
    attr['margin']           = margin
    attr['test_size']        = test_size
    attr['vector_limit']     = vector_limit
    attr['total_test']       = total_test_set
    attr['total_train']      = total_train_set
    attr['total_featureset'] = total_feature_sets
    attr.update(corpus_rules) # update taken from corpus rules


model_basename =␣
  ↪f"tfs({total_feature_sets})-vl({vector_limit})-m({margin})-{cr_name}"
save_report(classifiers, f"{model_basename}")
```

```
Saving Accuracy Level into file…
```

[ ]:
```
display_html('<h3>Display Accuracy History</h3><hr/>')
show_report_history(10)
```

```
                   name                                            trainer  \
0  Support Vector Machine  <bound method SklearnClassifier.train of <Skle…
1      Gradient Boosting  <bound method SklearnClassifier.train of <Skle…
2    Logistic Regression  <bound method SklearnClassifier.train of <Skle…
3     K-Nearest Neigbors  <bound method SklearnClassifier.train of <Skle…
4            Naive Bayes  <bound method NaiveBayesClassifier.train of <c…
5      Gradient Boosting  <bound method SklearnClassifier.train of <Skle…
6  Support Vector Machine  <bound method SklearnClassifier.train of <Skle…
7     K-Nearest Neigbors  <bound method SklearnClassifier.train of <Skle…
8            Naive Bayes  <bound method NaiveBayesClassifier.train of <c…
9    Logistic Regression  <bound method SklearnClassifier.train of <Skle…

                                      classifier            accuracy  \
0                        <SklearnClassifier(SVC())>  0.9613030397344123
1  <SklearnClassifier(GradientBoostingClassifier())>  0.9603693329183526
2          <SklearnClassifier(LogisticRegression())>  0.9569457412594667
3  <SklearnClassifier(KNeighborsClassifier(n_neig…  0.9525884427845213
4  <nltk.classify.naivebayes.NaiveBayesClassifier…  0.9520697167755992
5  <SklearnClassifier(GradientBoostingClassifier())>   0.944081336238199
6                        <SklearnClassifier(SVC())>   0.944081336238199
7  <SklearnClassifier(KNeighborsClassifier(n_neig…   0.944081336238199
8  <nltk.classify.naivebayes.NaiveBayesClassifier…  0.9394128021579002
9          <SklearnClassifier(LogisticRegression())>  0.9344330324722482
```

```
      elapse_time                                          challenges  \
0  0:00:52.648398  {'total_data': 8493, 'total_valid': 8238, 'tot…
1  0:00:11.014179  {'total_data': 8493, 'total_valid': 8269, 'tot…
2  0:00:05.454787  {'total_data': 8493, 'total_valid': 8155, 'tot…
3  0:02:11.603473  {'total_data': 8493, 'total_valid': 1043, 'tot…
4  0:00:05.225297  {'total_data': 8493, 'total_valid': 7518, 'tot…
5  0:00:03.653026  {'total_data': 8493, 'total_valid': 8181, 'tot…
6  0:00:23.812409  {'total_data': 8493, 'total_valid': 8181, 'tot…
7  0:00:40.372585  {'total_data': 8493, 'total_valid': 8181, 'tot…
8  0:00:01.110497  {'total_data': 8493, 'total_valid': 8078, 'tot…
9  0:00:01.276595  {'total_data': 8493, 'total_valid': 8078, 'tot…

                                   confussion_matrix  \
0  {'non-payload': {'label': 'non-payload', 'prec…
1  {'non-payload': {'label': 'non-payload', 'prec…
2  {'non-payload': {'label': 'non-payload', 'prec…
3  {'non-payload': {'label': 'non-payload', 'prec…
4  {'non-payload': {'label': 'non-payload', 'prec…
5  {'non-payload': {'label': 'non-payload', 'prec…
6  {'non-payload': {'label': 'non-payload', 'prec…
7  {'non-payload': {'label': 'non-payload', 'prec…
8  {'non-payload': {'label': 'non-payload', 'prec…
9  {'non-payload': {'label': 'non-payload', 'prec…

                     datetime margin test_size vector_limit total_test  \
0  2022-06-17 22:16:53.944323      3       0.2           72       9639
1  2022-06-17 22:16:53.944334      3       0.2           72       9639
2  2022-06-17 22:16:53.944336      3       0.2           72       9639
3  2022-06-17 22:16:53.944338      3       0.2           72       9639
4  2022-06-17 22:16:53.944340      3       0.2           72       9639
5  2022-06-17 21:27:52.481019    0.5       0.2           12       9639
6  2022-06-17 21:27:52.481028    0.5       0.2           12       9639
7  2022-06-17 21:27:52.481030    0.5       0.2           12       9639
8  2022-06-17 21:27:52.481031    0.5       0.2           12       9639
9  2022-06-17 21:27:52.481033    0.5       0.2           12       9639

  total_train total_featureset to_lowercase only_alphanum remove_puncts
0       38553            48192        False         False         False
1       38553            48192        False         False         False
2       38553            48192        False         False         False
3       38553            48192        False         False         False
4       38553            48192        False         False         False
5       38553            48192        False         False         False
6       38553            48192        False         False         False
7       38553            48192        False         False         False
8       38553            48192        False         False         False
9       38553            48192        False         False         False
```

[ ]:
```python
import pickle


save_classifier = True

if save_classifier:
    print("Saving classifier into file/object based...")
    for id, attr in classifiers.items():
        dirname = os.path.join(classifier_dir, model_basename)
        if not os.path.exists(dirname):
            os.makedirs(dirname)
        filename = os.path.join(dirname, f"{id}.model")
        with open(filename, "wb") as writer:
            pickle.dump(attr['classifier'], writer)


def load_classifier(name):
    filename = os.path.join(classifier_dir, f"{name}.model")
    if not os.path.exists(filename):
        raise ValueError(f"No classifier with name {name}")
    with open(filename, 'rb') as reader:
        classifier = pickle.load(reader)
        return classifier

print("All jobs finished...")
```

```
Saving classifier into file/object based…
All jobs finished…
```

SQLi-Notebook-Margin6.5

July 12, 2022

0.0.1 SQLi (SQL Injection) Attack Detection

SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. The machine learning model uses Python NLTK to build features and model. The use of Python NLTK will make the elaboration easier. In addition, scikit-learning module is also used to calculate some statistic calculation and comparison. It takes about 10 until 25 minutes to finish all jobs.

[ ]:
```
print('Installing required libraries...')
%pip install -q --upgrade pip
%pip install -q scikit-learn nltk wordcloud openpyxl python-slugify[unidecode]
print('Instalation done!')
```

```
Installing required libraries…
Note: you may need to restart the kernel to use updated packages.
Note: you may need to restart the kernel to use updated packages.
Instalation done!
```

[ ]:
```
import os
import csv
import json
import shutil
import pandas as panda
import numpy as np
import matplotlib.pyplot as plot
import nltk
from slugify import slugify
from nltk.tokenize import word_tokenize


# import sklearn functions
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
```

```
from IPython.display import display, display_html, HTML
from tqdm.notebook import tqdm_notebook
from wordcloud import WordCloud
from datetime import datetime
```

[ ]:
```
dataset_csv      = os.path.join('datasets', 'Dataset-SQLi-V3.csv')
challenge_csv    = os.path.join('datasets', 'Dataset-SQLi-V2.csv')

report_dir       = os.path.join('reports')
reporter_dir     = os.path.join(report_dir, 'json')
classifier_dir   = os.path.join('reports', 'classifiers')
corpus_dir       = os.path.join('output', 'corpus')

if not os.path.exists(reporter_dir):
    os.makedirs(reporter_dir)
```

[ ]:
```
df = panda.read_csv(dataset_csv)
df.sample(5)
```

[ ]:
```
                                                Sentence Label Unnamed: 2  \
7299        1 and elt  (  4249  =  4249,7259  )  -- meef     1        NaN
24128   SELECT * FROM per ORDER BY straight ASC, trai…     0        NaN
256     select * from users where id  =  '1' union se…     1        NaN
15637                            general castaos 180 9-e     0        NaN
15881                            ehrlich.selwyne@ebaz.ba     0        NaN

       Unnamed: 3
7299          NaN
24128         NaN
256           NaN
15637         NaN
15881         NaN
```

Cleaning and Positioning Dataset This step is for cleaning and positioning dataset, such as drop unused columns, remove empty values, and drop duplicate values.

[ ]:
```
data = df.drop(['Unnamed: 2', 'Unnamed: 3'], axis=1)
data['Label'] = panda.to_numeric(data['Label'], errors='coerce') # coerce will␣
 ↪make un digit become NaN
data.dropna(inplace=True) # remove NAN or empty values in the dataset
data.drop_duplicates(subset='Sentence') # drop duplicate values
data.sample(5)
```

[ ]:
```
                                                Sentence  Label
21945             SELECT DISTINCT recent FROM mysterious    0.0
904    1'  )  or 5286  =  (  select count  (  *  )…    1.0
3428   -2872  )  where 8797  =  8797 or 1 group by c…    1.0
```

```
8429   -6263%"   )    )    or 4747  =  dbms_utility.s…    1.0
12969                                              datario    0.0
```

[ ]:
```
# get more info about dataset which has been clean
rd_count = df[df.columns[0]].count() # real dataset
cd_count = data[data.columns[0]].count() # cleaned dataset

print("Total of original rows\t:", rd_count)
print("Total of eligible rows\t:", cd_count)
print("Total of removed rows\t:", rd_count - cd_count)
```

```
Total of original rows  : 30904
Total of eligible rows  : 30609
Total of removed rows   : 295
```

Convert CSV Dataset Format into File Based Corpus

[ ]:
```
# remove previous dataset for avoid error
if os.path.exists(corpus_dir):
    shutil.rmtree(corpus_dir)

pbar  = tqdm_notebook(total=len(data.values), desc="Convert CSV to corpus file")
index = 1
for payload, label in data.values:
    label     = 'payload' if label == 1 else 'non-payload'
    label_dir = os.path.join(corpus_dir, label)
    if not os.path.exists(label_dir):
        os.makedirs(label_dir)
    dt_filepath = os.path.join(label_dir, f'{index}.txt')
    with open(dt_filepath, 'w') as writer:
        writer.write(payload)
    pbar.update(1)
    index += 1
pbar.close()

del df, data
```

```
Convert CSV to corpus file:   0%|          | 0/30609 [00:00<?, ?it/s]
```

Insert payload corpus file into NLTK Corpus Reader

[ ]:
```
from nltk.corpus.reader import CategorizedPlaintextCorpusReader

corpus_set = CategorizedPlaintextCorpusReader(
    corpus_dir, r'(?!\.).*\.txt',
    cat_pattern=r'(payload|non-payload)/.*',
    encoding="utf-8"
)
```

```
print('Payload Total\t\t:',   len(corpus_set.fileids('payload')))
print('Non-Payload Total\t:', len(corpus_set.fileids('non-payload')))
print('Total Corpus\t\t:', len(corpus_set.fileids('payload')) + len(corpus_set.
  ↪fileids('non-payload')))
```

```
Payload Total           : 11341
Non-Payload Total       : 19268
Total Corpus            : 30609
```

Corpus Cleaning and Positioning

[ ]:
```
import string
import re, inspect

corpus_rules = {
    'to_lowercase'  : True,
    'only_alphanum' : False,
    'remove_puncts' : False,
}


def construct_corpus(corpus_in)->list:
    punctuations = list(string.punctuation)
    corpus_out   = []
    if type(corpus_in) == str:
        if corpus_rules['to_lowercase']:
            corpus_in = corpus_in.lower()
        corpus_in = word_tokenize(corpus_in)
    else:
        if corpus_rules['to_lowercase']:
            corpus_in = [w.lower() for w in corpus_in]
    for item in corpus_in:
        if not corpus_rules['remove_puncts']:
            punctuations = []
        if not item in punctuations:
            if corpus_rules['only_alphanum']:
                item = re.sub(r'[\W\d]', '', item)
            if item:
                corpus_out.append(item)
    return corpus_out


def construct_features(corpus:string)->dict:
    feature = {}
    corpus  = set(construct_corpus(corpus))
    for vector in feature_vectors.keys():
        feature[vector] = vector in corpus # corpus string -> corpus list
    return feature
```

```
def check_payload(classifier, payload:string):
    predict = classifier.classify(construct_features(payload))
    return predict


def get_accuracy(classifier, test_fs):
    predict = classifier.classify_many([fs for (fs, l) in test_fs])
    correct = [l == r for ((fs, l), r) in zip(test_fs, predict)]
    if correct:
        accuracy = sum(correct) / len(correct)
    else:
        accuracy = 0
    return [predict, accuracy]


def wordcloud_draw(data, filename, bgcolor='white'):
    words  = ' '.join(data)
    wCloud = WordCloud(background_color=bgcolor, width=2250, height=1000).
 ↪generate(words)
    wCloud.to_file(filename)
    plot.figure(1, figsize=(20, 10))
    plot.imshow(wCloud)
    plot.axis('off')
    plot.show()


def wordplot_draw(freq_words, filename, graph_limit=30):
    fig = plot.figure(figsize=(16, 7))
    plot.gcf().subplots_adjust(bottom=0.15)
    freq_words.plot(graph_limit, title=f'{graph_limit} Most Common Words')
    plot.show()
    plot.draw()
    fig.savefig(filename, dpi=120)


def save_report(classifiers, report_name):
    filename = os.path.join(reporter_dir, f"{report_name}.json")
    reports  = []
    for id, attr in classifiers.items():
        item = {k:str(v) for k,v in attr.items()}
        reports.append(item)
    if not reports:
        return None
    with open(filename, 'w') as writer:
        json.dump(reports, writer)
```

```
def get_report_history():
    reports = []
    for report_file in os.listdir(reporter_dir):
        filename = os.path.join(reporter_dir, report_file)
        if filename.endswith('.json'):
            with open(filename) as reader:
                report   = json.load(reader)
                reports += report
    return reports


def show_report_history(n=10):
    reports = get_report_history()
    if reports:
        tf = panda.DataFrame.from_records([item.values() for item in reports],␣
 ↪columns=[list(reports[0].keys())])
        filepath  = os.path.join(os.getcwd(), report_dir, 'report')
        tf.to_excel(f"{filepath}.xlsx")
        tf.to_html(f"{filepath}.html")
        display(tf.head(n))
```

[ ]:
```
pbar = tqdm_notebook(corpus_set.fileids(), desc="Calculating words count")
words_counter = []
for file_id in corpus_set.fileids():
    word_count = len(corpus_set.words(fileids=file_id))
    words_counter.append(word_count)
    pbar.update(1)
pbar.close()

margin       = 6.5
vector_limit = int((sum(words_counter)/len(words_counter)) * margin)
print(f'Vector limit is {vector_limit}')
```

```
Calculating words count:   0%|          | 0/30609 [00:00<?, ?it/s]

Vector limit is 104
```

[ ]:
```
feature_vectors = {}
_freq_words     = {} # for informational purpose

for category in corpus_set.categories():
    print(f"Injecting {vector_limit} {category} vectors to feature vectors...")
    corpus_cat = corpus_set.words(categories=[category])
    freq_words = nltk.FreqDist(construct_corpus(corpus_cat))
```

```
    most_words = {word.lower():n for word, n in freq_words.
 ↪most_common(vector_limit)}
    _freq_words[category] = [most_words, freq_words]
    feature_vectors.update(most_words)

    t = {f"{word.lower()},{n}" for word, n in freq_words.most_common(50)}
    display(t)


# save feature fectors
filename = f"feature_vector-tfs({len(corpus_set.
 ↪fileids())})-vl({vector_limit})-m({margin})"
cr_name  = '-'.join([f"{k}({int(v)})" for k,v in corpus_rules.items()])
with open(os.path.join(classifier_dir, f'{filename}-{cr_name}.json'), "w") as␣
 ↪writer:
    json.dump(feature_vectors, writer)
    print('Feature vector saved...')
```

```
Injecting 104 non-payload vectors to feature vectors…
```

```
{'",845',
 "',,1540",
 "',10468",
 '(,5540',
 '),5430',
 '*,6356',
 '+,795',
 ',,4893',
 '-,1468',
 '.,3436',
 '/,773',
 '01,332',
 '07,430',
 '1996,428',
 '3,1543',
 '31,326',
 '50,735',
 ';,327',
 '=,4034',
 '@,911',
 'and,1354',
 'as,1304',
 'avg,745',
 'between,870',
 'by,1213',
 'count,894',
 'employees,663',
 'fetch,837',
```

```
 'first,859',
 'from,11860',
 'id,332',
 'in,557',
 'join,1034',
 'like,452',
 'limit,390',
 'not,1104',
 'on,496',
 'only,636',
 'or,355',
 'order,1087',
 'orders,813',
 'percent,747',
 'rows,837',
 's,1288',
 'select,12496',
 'sum,449',
 'top,979',
 'union,563',
 'where,4308',
 'wp_posts,392'}

Injecting 104 payload vectors to feature vectors…

{'",5148',
 '#,1938',
 '$,1058',
 "',9440",
 '(,40095',
 '),45252',
 '*,3356',
 '+,2205',
 ',,20265',
 '-,2413',
 '--,3875',
 '.,2401',
 '0,2224',
 '0x7171706a71,783',
 '1,11506',
 '106,807',
 '112,865',
 '113,3755',
 '122,1502',
 '5,1442',
 '=,13701',
 'all,1849',
 'all_users,1134',
 'and,6024',
```

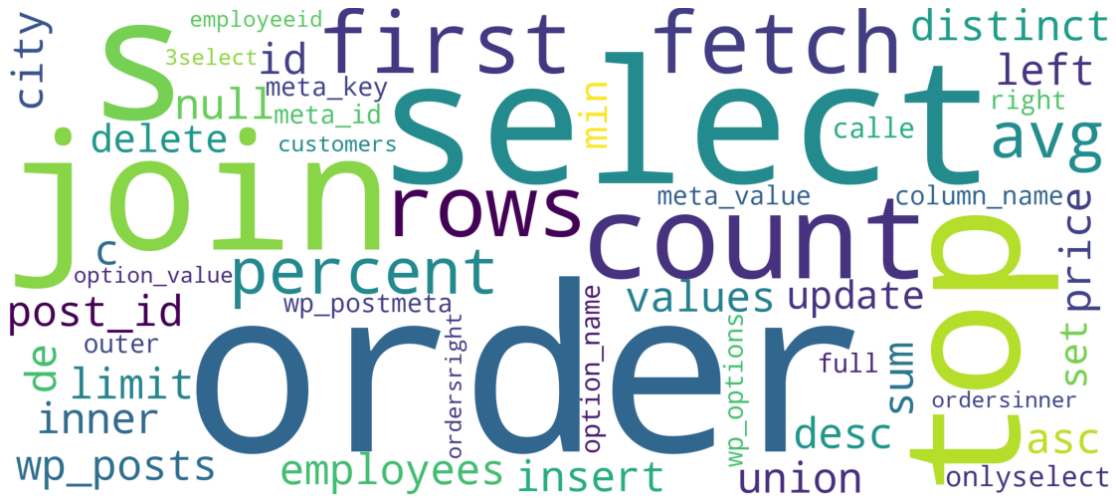
```
 'as,5109',
 'case,1586',
 'char,5629',
 'chr,6588',
 'concat,875',
 'count,1536',
 'dual,895',
 'else,1629',
 'elt,1078',
 'end,1651',
 'from,4297',
 'like,1032',
 'null,5022',
 'or,3564',
 'rdb,1001',
 'select,9999',
 'sleep,872',
 'sysusers,1309',
 't1,894',
 't2,894',
 't3,894',
 'then,1589',
 'union,2442',
 'when,1586',
 'where,4069',
 '||,8367'}
```

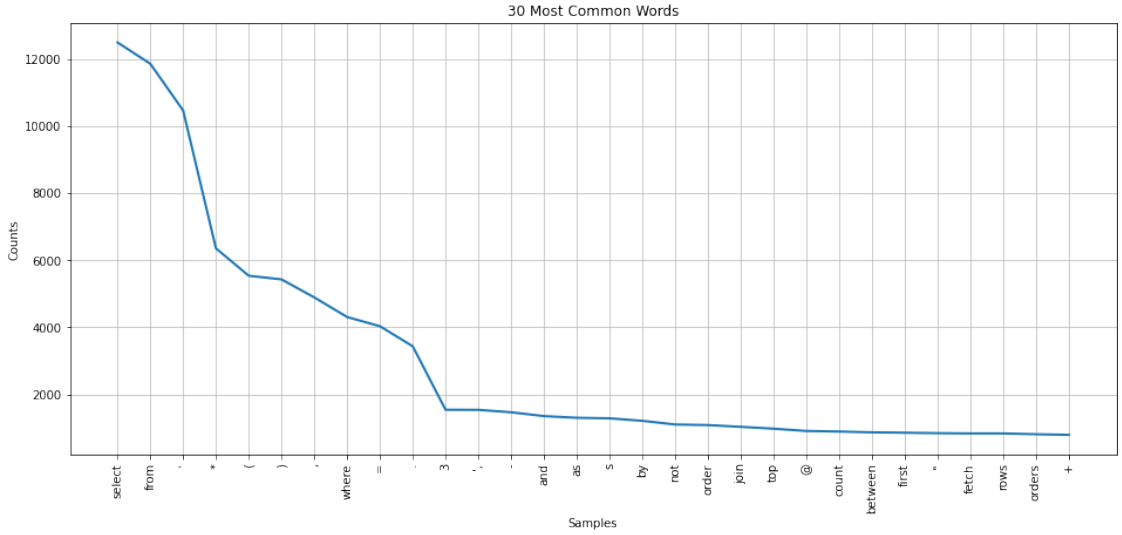
```
Feature vector saved…
```

Informational Purpose

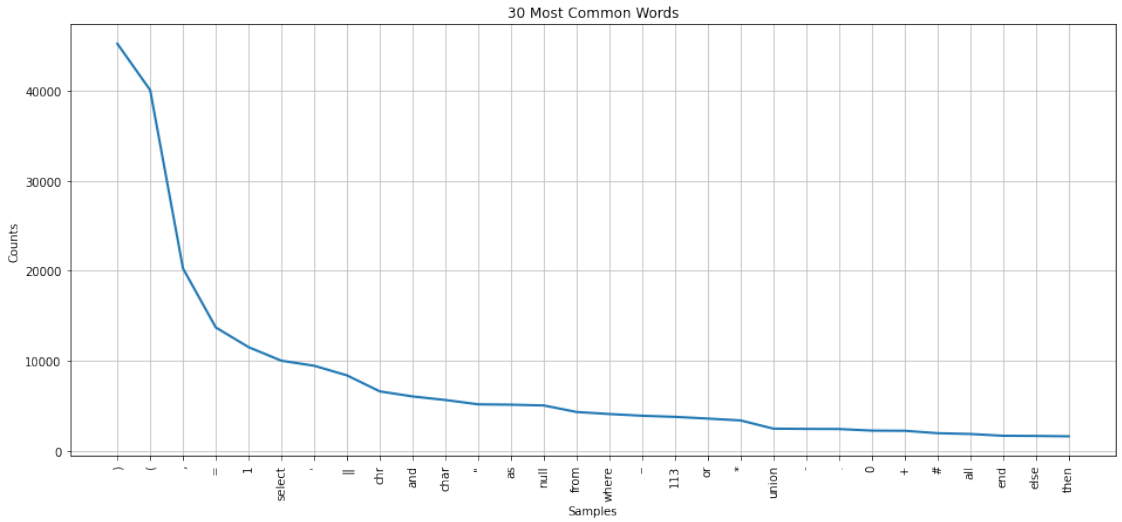
[ ]:
```
graph_limit = 30
figure_dir  = os.path.join(report_dir, 'figures')
if not os.path.exists(figure_dir):
    os.makedirs(figure_dir)

for category, attrs in _freq_words.items():
    print(f'Showing {vector_limit} most common words in {category}...')
    most_word, freq_words = attrs
    filename = f"{category}-tfs({len(corpus_set.
 ↪fileids())})-vl({vector_limit})-m({margin})"
    cr_name  = '-'.join([f"{k}({int(v)})" for k,v in corpus_rules.items()])
    wordcloud_draw(most_word, os.path.join(figure_dir,␣
 ↪f'WC-{filename}-{cr_name}.png'))
    wordplot_draw(freq_words, os.path.join(figure_dir,␣
 ↪f'WP-{filename}-{cr_name}.png'), graph_limit)

del _freq_words
```

Showing 104 most common words in non-payload…





Showing 104 most common words in payload…



<Figure size 432x288 with 0 Axes>

Preparing Feature Enginering Functions

[ ]:
```
feature_set_dir = os.path.join('output', 'feature-sets')

# remove previous feature_set_dir for avoid error
if os.path.exists(feature_set_dir):
    print('Removing old feature-sets...')
    shutil.rmtree(feature_set_dir)

if not os.path.exists(feature_set_dir):
    os.makedirs(feature_set_dir)
```

```
datasets  = []
pbar      = tqdm_notebook(corpus_set.fileids())
step      = 1
for category in corpus_set.categories():
    pbar.set_description(f"({step}/{len(corpus_set.categories())}) Classifying␣
 ↪Courpus -> {category}")
    labels = corpus_set.fileids(category)
    for file_id in labels:
        content = corpus_set.raw(file_id)
        datasets.append((file_id, content, category))
        pbar.update(1)
    step += 1
pbar.close()

del corpus_set
```

```
Removing old feature-sets…

  0%|          | 0/30609 [00:00<?, ?it/s]
```

[ ]:
```
feature_sets = []
pbar         = tqdm_notebook(total=len(datasets), desc=f"Processing Feature Set␣
 ↪with {vector_limit} Vector Limit")
for (file_id, payloads, category) in datasets:
    basename = os.path.basename(file_id)
    filename = os.path.join(feature_set_dir, basename)
    features = construct_features(payloads)
    # with open(filename, 'w') as writer: json.dump([features, category],␣
 ↪writer)
    feature_sets.append([features, category])
    pbar.update(1)
pbar.close()

del datasets
```

```
Processing Feature Set with 104 Vector Limit:   0%|          | 0/30609 [00:00<?,␣
 ↪?it/s]
```

Spliting Data Training and Testing

[ ]:
```
from sklearn import model_selection


test_size = 0.2 # percentation of data testing
X, y      = [x for x,y in feature_sets], [y for x,y in feature_sets]

# split data training and data testing using Python NLTK
```

```
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y,␣
 ↪test_size=test_size)

# melihat jumlah data training dan testing
total_train_set     = len(X_train)
total_test_set      = len(X_test)
total_feature_sets  = len(feature_sets)

print('Data Training\t:', total_train_set)
print('Data Testing \t:', total_test_set)

del feature_sets
```

```
Data Training   : 24487
Data Testing    : 6122
```

Training or Fitting Dataset using Classifier

[ ]:
```
from nltk import classify

scikitNLTK          = classify.scikitlearn.SklearnClassifier
classifiers         = {
    'naive-bayes'   : {
        'name'      : 'Naive Bayes',
        'trainer'   : classify.NaiveBayesClassifier.train,
    },
    'logistic-regress': {
        'name'      : 'Logistic Regression',
        'trainer'   : scikitNLTK(LogisticRegression()).train,
    },
    'gradient-boost': {
        'name'      : 'Gradient Boosting',
        'trainer'   : scikitNLTK(GradientBoostingClassifier()).train,
    },
    'sv-machine': {
        'name'      : 'Support Vector Machine',
        'trainer'   : scikitNLTK(SVC()).train,
    },
    'kn-neighbors'  : {
        'name'      : 'K-Nearest Neigbors',
        'trainer'   : scikitNLTK(KNeighborsClassifier(n_neighbors=3)).train,
    },
}
```

Testing or Getting Accuracy Level of Classifier

[ ]:
```
print(f"Training information -> vector_limit {vector_limit} | margin: {margin}␣
 ↪| test_size: {test_size}...")
```

```
train_featureset    = [[X_train[i], y_train[i]] for i in range(total_train_set)]
test_featureset     = [[X_test[i], y_test[i]] for i in range(total_test_set)]

counter = 1
for id, attr in classifiers.items():
    print(f"[{counter}] Training data using {attr['name']}...")
    start_time          = datetime.now()
    attr['classifier']  = attr['trainer'](train_featureset)
    attr['y_predict'], attr['accuracy'] = get_accuracy(attr['classifier'],␣
 ↪test_featureset)
    attr['elapse_time'] = str(datetime.now() - start_time)
    print(f"Accuracy Level\t: {attr['accuracy']}")
    print(f"Time Ellapsed\t: {attr['elapse_time']}\n")
    counter += 1


print(f'Ordering classifier based on accuracy level...')
index        = 1
classifiers  = {k: v for k, v in sorted(classifiers.items(), key=lambda item:␣
 ↪item[1]['accuracy'], reverse=True)}
for id, attr in classifiers.items():
    print(f"[{index}] {attr['name']} (Accuracy: {attr['accuracy']}) (Time␣
 ↪Ellapse: {attr['elapse_time']})")
    index += 1


del train_featureset, test_featureset
```

```
Training information -> vector_limit 104 | margin: 6.5 | test_size: 0.2…
[1] Training data using Naive Bayes…
Accuracy Level  : 0.9754982032015681
Time Ellapsed   : 0:00:04.138092

[2] Training data using Logistic Regression…
Accuracy Level  : 0.9960797125122509
Time Ellapsed   : 0:00:04.485225

[3] Training data using Gradient Boosting…
Accuracy Level  : 0.9947729500163345
Time Ellapsed   : 0:00:13.379335

[4] Training data using Support Vector Machine…
Accuracy Level  : 0.9977131656321464
Time Ellapsed   : 0:00:10.713473

[5] Training data using K-Nearest Neigbors…
Accuracy Level  : 0.9970597843841882
```

```
Time Ellapsed   : 0:01:23.933334

Ordering classifier based on accuracy level…
[1] Support Vector Machine (Accuracy: 0.9977131656321464) (Time Ellapse:
0:00:10.713473)
[2] K-Nearest Neigbors (Accuracy: 0.9970597843841882) (Time Ellapse:
0:01:23.933334)
[3] Logistic Regression (Accuracy: 0.9960797125122509) (Time Ellapse:
0:00:04.485225)
[4] Gradient Boosting (Accuracy: 0.9947729500163345) (Time Ellapse:
0:00:13.379335)
[5] Naive Bayes (Accuracy: 0.9754982032015681) (Time Ellapse: 0:00:04.138092)
```

Observable/Simple Evaluation and Optimalization of Classifier/Model

[ ]:
```
custom_payloads = [
    ("<>Ini teks biasa yang bukan payload <h1>judul</h1>", 0),
    ("<h1>Teks bukan payload</h1>", 0),
    ("select // any Wxkxkxkxk wkwkwkw", 0),
    ('select * from users where id  =  1 or 1#"  (   union select 1,version  ( ␣
 ↪ )   -- 1', 1),
    ("SELECT * FROM Users WHERE UserId = 105 OR 1=1;", 1),
    ("SELECT UserId, Nama, Password FROM Users WHERE UserId = 105 or 1=1;", 1),
    ("SELECT * FROM Users WHERE UserId = 105; DROP TABLE Suppliers;", 1),
]
```

[ ]:
```
for id, attr in classifiers.items():
    table_row, tracks = [], []
    for payload, label in custom_payloads:
        label      = 'payload' if label == 1 else 'non-payload'
        prediction = check_payload(attr['classifier'], payload)
        status     = label == prediction
        table_row.append([payload, label, prediction, status])
        tracks.append(status)
    table_frame = panda.DataFrame.from_records(table_row, columns=['Payload',␣
 ↪'Actual', 'Predict', 'Status'])
    valid_preds = []
    for status in tracks:
        if status is True: valid_preds.append(status)
    display_html(f"<h3>{attr['name']} Prediction Result ({len(valid_preds)}/
 ↪{len(tracks)})</h3><hr/>", raw=True)
    display(table_frame)
```

```
                                          Payload       Actual  \
0  <>Ini teks biasa yang bukan payload <h1>judul<…  non-payload
1                       <h1>Teks bukan payload</h1>  non-payload
2                    select // any Wxkxkxkxk wkwkwkw  non-payload
3  select * from users where id  =  1 or 1#"  (  …      payload
```

```
4     SELECT * FROM Users WHERE UserId = 105 OR 1=1;      payload
5  SELECT UserId, Nama, Password FROM Users WHERE…      payload
6  SELECT * FROM Users WHERE UserId = 105; DROP T…      payload

       Predict  Status
0  non-payload    True
1  non-payload    True
2  non-payload    True
3      payload    True
4  non-payload   False
5  non-payload   False
6  non-payload   False
                                            Payload       Actual  \
0  <>Ini teks biasa yang bukan payload <h1>judul<…  non-payload
1                        <h1>Teks bukan payload</h1>  non-payload
2                    select // any Wxkxkxkxk wkwkwkw  non-payload
3  select * from users where id  =  1 or 1#"  (  …      payload
4     SELECT * FROM Users WHERE UserId = 105 OR 1=1;      payload
5  SELECT UserId, Nama, Password FROM Users WHERE…      payload
6  SELECT * FROM Users WHERE UserId = 105; DROP T…      payload

       Predict  Status
0  non-payload    True
1  non-payload    True
2  non-payload    True
3      payload    True
4  non-payload   False
5  non-payload   False
6  non-payload   False
                                            Payload       Actual  \
0  <>Ini teks biasa yang bukan payload <h1>judul<…  non-payload
1                        <h1>Teks bukan payload</h1>  non-payload
2                    select // any Wxkxkxkxk wkwkwkw  non-payload
3  select * from users where id  =  1 or 1#"  (  …      payload
4     SELECT * FROM Users WHERE UserId = 105 OR 1=1;      payload
5  SELECT UserId, Nama, Password FROM Users WHERE…      payload
6  SELECT * FROM Users WHERE UserId = 105; DROP T…      payload

       Predict  Status
0  non-payload    True
1  non-payload    True
2  non-payload    True
3      payload    True
4  non-payload   False
5  non-payload   False
6  non-payload   False
```

```
                                              Payload       Actual  \
0  <>Ini teks biasa yang bukan payload <h1>judul<…  non-payload
1                        <h1>Teks bukan payload</h1>  non-payload
2                    select // any Wxkxkxkxk wkwkwkw  non-payload
3  select * from users where id  =  1 or 1#"  (  …      payload
4     SELECT * FROM Users WHERE UserId = 105 OR 1=1;      payload
5  SELECT UserId, Nama, Password FROM Users WHERE…      payload
6  SELECT * FROM Users WHERE UserId = 105; DROP T…      payload

       Predict  Status
0  non-payload    True
1  non-payload    True
2  non-payload    True
3      payload    True
4  non-payload   False
5  non-payload   False
6  non-payload   False
```

```
                                              Payload       Actual  \
0  <>Ini teks biasa yang bukan payload <h1>judul<…  non-payload
1                        <h1>Teks bukan payload</h1>  non-payload
2                    select // any Wxkxkxkxk wkwkwkw  non-payload
3  select * from users where id  =  1 or 1#"  (  …      payload
4     SELECT * FROM Users WHERE UserId = 105 OR 1=1;      payload
5  SELECT UserId, Nama, Password FROM Users WHERE…      payload
6  SELECT * FROM Users WHERE UserId = 105; DROP T…      payload

       Predict  Status
0  non-payload    True
1  non-payload    True
2  non-payload    True
3      payload    True
4      payload    True
5      payload    True
6  non-payload   False
```

Advanced Evaluation and Testing of Classifier

[ ]:
```
df_challenge          = panda.read_csv(challenge_csv, encoding='utf-16')
df_challenge['Label'] = panda.to_numeric(df_challenge['Label'])
df_challenge.drop_duplicates(inplace=True) # drop duplicate
df_challenge.dropna(subset='Sentence', inplace=True) # drop empty values
display_html(f"<h3>Total data will be predicted by classifier:␣
 ↪{len(df_challenge.values)}</h3><hr/>", raw=True)

df_challenge.sample(n=5)
```

[ ]:
```
                                                  Sentence  Label
448     select * from users where id = 1 union select…      1
32524                                  5657119326927439      0
2626                                        Critics Sen      0
19519                                           yu-hung      0
13796  1 )  where 5464 = 5464 or 1022 =  ( select cou…      1
```

[ ]:
```
step = 1
pbar = tqdm_notebook()
for id, attr in classifiers.items():
    pbar.reset(total=len(df_challenge.values))
    pbar.set_description_str(f"({step}/{len(classifiers)}) Predicting with␣
 ↪{attr['name']}")
    tracks     = []
    start_time = datetime.now()
    for payload, label in df_challenge.values:
        label      = 'payload' if label == 1 else 'non-payload'
        prediction = check_payload(attr['classifier'], payload)
        status     = label == prediction
        tracks.append(status)
        pbar.update(1)
    valid_preds = []
    for status in tracks:
        if status is True: valid_preds.append(status)

    ellapse_time        = str(datetime.now() - start_time)
    attr['challenges']  = {
        'total_data'    : len(tracks),
        'total_valid'   : len(valid_preds),
        'total_invalid' : (len(tracks) - len(valid_preds)),
        'ellapse_time'  : ellapse_time,
    }
    percentation = int((len(valid_preds)/len(tracks)) * 100)
    information  = ' | '.join([f"{k.capitalize()}: {v}" for k,v in␣
 ↪attr['challenges'].items()])
    display_html(f"<div>{attr['name']} Prediction ({percentation}%)</div>",␣
 ↪raw=True)
    display_html(f"<div>{information}<hr/></div>", raw=True)
    step += 1
pbar.close()

total_payload_challenge = len(df_challenge)
del df_challenge
```

```
0it [00:00, ?it/s]
```

Confussion Matrix of Classifier Based on Highest Accuracy

[ ]:
```
def get_confussion_matrix(TP, FP, FN):
    confussion_matrix = {}
    for label in sorted(labels):
        precision, recall = 0, 0
        if TP[label] == 0:
            f_measure = 0
        else:
            precision = float(TP[label]) / (TP[label] + FP[label])
            recall    = float(TP[label]) / (TP[label] + FN[label])
            f_measure = float(2) * (precision * recall) / (precision + recall)
        confussion_matrix[label] = {
            'label'     : label,
            'precision' : precision,
            'recall'    : recall,
            'f_measure' : f_measure,
        }
    return confussion_matrix


def table_confussion_matrix(confussion_matrix):
    table_rows         = [v.values() for k,v in confussion_matrix.items()]
    table_frame        = panda.DataFrame.from_records(table_rows,␣
 ↪columns=['Label', 'Precision', 'Recall', 'F-Measure'])
    return table_frame
```

[ ]:
```
from collections import Counter

step = 1
for id, attr in classifiers.items():
    nltk_cm = nltk.ConfusionMatrix(y_test, attr['y_predict'], True)
    labels  = {'payload', 'non-payload'}

    TP, FN, FP = Counter(), Counter(), Counter()
    for i in labels:
        for j in labels:
            if i == j:
                TP[i] += int(nltk_cm[i,j])
            else:
                FN[i] += int(nltk_cm[i,j])
                FP[j] += int(nltk_cm[i,j])

    confussion_matrix         = get_confussion_matrix(TP, FP, FN)
    attr['confussion_matrix'] = confussion_matrix
    print(nltk_cm.pretty_format())
    display(table_confussion_matrix(confussion_matrix))
    print(f"With accuracy level -> {attr['accuracy']}")
    display_html('<hr/>', raw=True)
```

```
    step += 1

    # remove unused variable
    del attr['y_predict']
```

```
            |    n      |
            |    o      |
            |    n      |
            |    -      |
            |    p    p |
            |    a    a |
            |    y    y |
            |    l    l |
            |    o    o |
            |    a    a |
            |    d    d |
------------+-----------+
non-payload |<3890>   1 |
    payload |   13<2218>|
------------+-----------+
(row = reference; col = test)
```

```
         Label  Precision    Recall  F-Measure
0  non-payload   0.996669  0.999743   0.998204
1      payload   0.999549  0.994173   0.996854
```

```
With accuracy level -> 0.9977131656321464
```

```
            |    n      |
            |    o      |
            |    n      |
            |    -      |
            |    p    p |
            |    a    a |
            |    y    y |
            |    l    l |
            |    o    o |
            |    a    a |
            |    d    d |
------------+-----------+
non-payload |<3887>   4 |
    payload |   14<2217>|
------------+-----------+
(row = reference; col = test)
```

```
         Label  Precision    Recall  F-Measure
0  non-payload   0.996411  0.998972   0.997690
```

```
1      payload   0.998199  0.993725   0.995957
```

```
With accuracy level -> 0.9970597843841882
```

```
             |    n      |
             |    o      |
             |    n      |
             |    -      |
             |    p    p |
             |    a    a |
             |    y    y |
             |    l    l |
             |    o    o |
             |    a    a |
             |    d    d |
-------------+-----------+
non-payload |<3889>   2 |
    payload |   22<2209>|
-------------+-----------+
(row = reference; col = test)
```

```
         Label  Precision    Recall  F-Measure
0  non-payload   0.994375  0.999486   0.996924
1      payload   0.999095  0.990139   0.994597
```

```
With accuracy level -> 0.9960797125122509
```

```
             |    n      |
             |    o      |
             |    n      |
             |    -      |
             |    p    p |
             |    a    a |
             |    y    y |
             |    l    l |
             |    o    o |
             |    a    a |
             |    d    d |
-------------+-----------+
non-payload |<3887>   4 |
    payload |   28<2203>|
-------------+-----------+
(row = reference; col = test)
```

```
         Label  Precision    Recall  F-Measure
0  non-payload   0.992848  0.998972   0.995901
1      payload   0.998188  0.987450   0.992790
```

```
With accuracy level -> 0.9947729500163345
```

```
             |    n      |
             |    o      |
             |    n      |
             |    -      |
             |    p    p |
             |    a    a |
             |    y    y |
             |    l    l |
             |    o    o |
             |    a    a |
             |    d    d |
-------------+-----------+
non-payload |<3859>  32 |
    payload |  118<2113>|
-------------+-----------+
(row = reference; col = test)
```

	Label	Precision	Recall	F-Measure
0	non-payload	0.970329	0.991776	0.980935
1	payload	0.985082	0.947109	0.965722

With accuracy level -> 0.9754982032015681

[ ]:
```
print('Saving Accuracy Level into file...')

for id, attr in classifiers.items():
    attr['datetime']         = datetime.now()
    attr['margin']           = margin
    attr['test_size']        = test_size
    attr['vector_limit']     = vector_limit
    attr['total_test']       = total_test_set
    attr['total_train']      = total_train_set
    attr['total_featureset'] = total_feature_sets
    attr.update(corpus_rules) # update taken from corpus rules


model_basename =␣
  ↪f"tfs({total_feature_sets})-vl({vector_limit})-m({margin})-{cr_name}"
save_report(classifiers, f"{model_basename}")
```

Saving Accuracy Level into file…

[ ]:
```
display_html('<h3>Display Accuracy History</h3><hr/>')
show_report_history(10)
```

	name	trainer	\
0	Support Vector Machine	<bound method SklearnClassifier.train of <Skle…	
1	K-Nearest Neigbors	<bound method SklearnClassifier.train of <Skle…	

```
2       Gradient Boosting  <bound method SklearnClassifier.train of <Skle…
3     Logistic Regression  <bound method SklearnClassifier.train of <Skle…
4             Naive Bayes  <bound method NaiveBayesClassifier.train of <c…
5  Support Vector Machine  <bound method SklearnClassifier.train of <Skle…
6      K-Nearest Neigbors  <bound method SklearnClassifier.train of <Skle…
7       Gradient Boosting  <bound method SklearnClassifier.train of <Skle…
8     Logistic Regression  <bound method SklearnClassifier.train of <Skle…
9             Naive Bayes  <bound method NaiveBayesClassifier.train of <c…

                                      classifier            accuracy  \
0                        <SklearnClassifier(SVC())>    0.99640640313623
1  <SklearnClassifier(KNeighborsClassifier(n_neig…  0.9962430578242405
2  <SklearnClassifier(GradientBoostingClassifier())>  0.9916693890885332
3          <SklearnClassifier(LogisticRegression())>  0.9725579875857563
4  <nltk.classify.naivebayes.NaiveBayesClassifier…  0.9429924861156485
5                        <SklearnClassifier(SVC())>  0.9647174126102581
6  <SklearnClassifier(KNeighborsClassifier(n_neig…   0.962593923554394
7  <SklearnClassifier(GradientBoostingClassifier())>   0.960797125122509
8          <SklearnClassifier(LogisticRegression())>  0.9294348252205161
9  <nltk.classify.naivebayes.NaiveBayesClassifier…  0.8820646847435478

      elapse_time                                          challenges  \
0  0:00:02.501898  {'total_data': 33727, 'total_valid': 33376, 't…
1  0:00:12.989578  {'total_data': 33727, 'total_valid': 33417, 't…
2  0:00:03.443110  {'total_data': 33727, 'total_valid': 33398, 't…
3  0:00:00.956156  {'total_data': 33727, 'total_valid': 33216, 't…
4  0:00:00.756524  {'total_data': 33727, 'total_valid': 32403, 't…
5  0:00:07.304314  {'total_data': 33727, 'total_valid': 32990, 't…
6  0:00:19.023528  {'total_data': 33727, 'total_valid': 33069, 't…
7  0:00:02.731510  {'total_data': 33727, 'total_valid': 33042, 't…
8  0:00:00.741008  {'total_data': 33727, 'total_valid': 31965, 't…
9  0:00:00.570328  {'total_data': 33727, 'total_valid': 30636, 't…

                                   confussion_matrix  \
0  {'non-payload': {'label': 'non-payload', 'prec…
1  {'non-payload': {'label': 'non-payload', 'prec…
2  {'non-payload': {'label': 'non-payload', 'prec…
3  {'non-payload': {'label': 'non-payload', 'prec…
4  {'non-payload': {'label': 'non-payload', 'prec…
5  {'non-payload': {'label': 'non-payload', 'prec…
6  {'non-payload': {'label': 'non-payload', 'prec…
7  {'non-payload': {'label': 'non-payload', 'prec…
8  {'non-payload': {'label': 'non-payload', 'prec…
9  {'non-payload': {'label': 'non-payload', 'prec…

                     datetime margin test_size vector_limit total_test  \
0  2022-06-20 09:24:37.655268    1.5       0.2           24       6122
1  2022-06-20 09:24:37.655276    1.5       0.2           24       6122
```

```
2  2022-06-20 09:24:37.655278    1.5       0.2           24       6122
3  2022-06-20 09:24:37.655280    1.5       0.2           24       6122
4  2022-06-20 09:24:37.655281    1.5       0.2           24       6122
5  2022-06-17 21:54:43.545851      1       0.2           16       6122
6  2022-06-17 21:54:43.545861      1       0.2           16       6122
7  2022-06-17 21:54:43.545863      1       0.2           16       6122
8  2022-06-17 21:54:43.545866      1       0.2           16       6122
9  2022-06-17 21:54:43.545868      1       0.2           16       6122

  total_train total_featureset to_lowercase only_alphanum remove_puncts
0       24487            30609         True         False         False
1       24487            30609         True         False         False
2       24487            30609         True         False         False
3       24487            30609         True         False         False
4       24487            30609         True         False         False
5       24487            30609         True          True          True
6       24487            30609         True          True          True
7       24487            30609         True          True          True
8       24487            30609         True          True          True
9       24487            30609         True          True          True
```

[ ]:
```python
import pickle

save_classifier = True

if save_classifier:
    print("Saving classifier into file/object based...")
    for id, attr in classifiers.items():
        dirname = os.path.join(classifier_dir, model_basename)
        if not os.path.exists(dirname):
            os.makedirs(dirname)
        filename = os.path.join(dirname, f"{id}.model")
        with open(filename, "wb") as writer:
            pickle.dump(attr['classifier'], writer)


def load_classifier(name):
    filename = os.path.join(classifier_dir, f"{name}.model")
    if not os.path.exists(filename):
        raise ValueError(f"No classifier with name {name}")
    with open(filename, 'rb') as reader:
        classifier = pickle.load(reader)
        return classifier

print("All jobs finished...")
```

```
Saving classifier into file/object based…
All jobs finished…
```

XSS-Notebook-Margin5

July 7, 2022

0.0.1 XSS (Cross Site Scripting) Attack Detection

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. The machine learning model uses Python NLTK to build features and model. The use of Python NLTK will make the elaboration easier. In addition, scikit-learning module is also used to calculate some statistic calculation and comparison. It takes about 10 until 25 minutes to finish all jobs.

1. Cross site scripting main dataset (Main Dataset).
2. Cross site scripting (XSS) dataset for deep learning (Challenging Dataset).
3. Cross site scripting (XSS) vulnerability payload list (Supportive Dataset).

[ ]:
```
print('Installing required libraries...')
%pip install -q --upgrade pip
%pip install -q scikit-learn nltk wordcloud openpyxl python-slugify[unidecode]
print('Instalation done!')
```

```
Installing required libraries…
Note: you may need to restart the kernel to use updated packages.
Note: you may need to restart the kernel to use updated packages.
Instalation done!
```

[ ]:
```
import os
import csv
import json
import shutil
import pandas as panda
import numpy as np
import matplotlib.pyplot as plot
import nltk
from slugify import slugify
from nltk.tokenize import word_tokenize


# import sklearn functions
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC

from IPython.display import display, display_html, HTML
from tqdm.notebook import tqdm_notebook
from wordcloud import WordCloud
from datetime import datetime
```

[ ]:
```
# download required files for building features
nltk.download('stopwords')
nltk.download('punkt')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/joearton/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /home/joearton/nltk_data…
[nltk_data]   Package punkt is already up-to-date!
```

[ ]:
```
True
```

[ ]:
```
dataset_csv1    = os.path.join('datasets', 'Dataset-XSS-Primary.csv') # https://
 ↪github.com/fmereani/Cross-Site-Scripting-XSS
dataset_csv2    = os.path.join('datasets', 'Dataset-XSS-Secondary.csv') # https:
 ↪//github.com/payloadbox/xss-payload-list
challenge_csv   = os.path.join('datasets', 'Dataset-XSS-Challenge.csv') # https:
 ↪//www.kaggle.com/datasets/syedsaqlainhussain/
 ↪cross-site-scripting-xss-dataset-for-deep-learning

report_dir      = os.path.join('reports')
reporter_dir    = os.path.join(report_dir, 'json')
classifier_dir  = os.path.join('reports', 'classifiers')
corpus_dir      = os.path.join('output', 'corpus')

if not os.path.exists(reporter_dir):
    os.makedirs(reporter_dir)
```

Preparing Dataset and Compilation

[ ]:
```
print('Content of Dataset 1 (Main Dataset)')
df1 = panda.read_csv(dataset_csv1)
display(df1.sample(5))

print('Content of Dataset 2 (Supportive Dataset)')
df2 = panda.read_csv(dataset_csv2)
display(df1.sample(5))
```

```
Content of Dataset 1 (Main Dataset)
```

```
                                                  Sentence  Label
23199  you can do better than this liked -price - app…      0
20189  loved the biltmore a most fascinating hotel an…      0
15072  http://www.wikihow.com/say-goodbye-to-bad-smel…      0
33513  http://www.wikihow.com/break-an-end-crystal-in…      0
20022  best service, great location. i stay in la oft…      0

Content of Dataset 2 (Supportive Dataset)

                                                  Sentence  Label
30635  http://www.bmzg.net/search.asp?q=http%3a%2f%2f…      1
2298   http://www.lorraine.ecologie.gouv.fr/spip.php?…      1
11195  http://localhost:8080/tienda1/publico/autentic…      0
26103  /* aerogear javascript library* https://github…      0
21439              ganz ok showreview(17443980, 'full');      0
```

Cleaning and Positioning Dataset This step is for cleaning and positioning dataset, such as drop unused columns, remove empty values, and drop duplicate values.

[ ]:
```
# combining df1 and df 2
frames = [df1, df2]
df     = panda.concat(frames)

# remove NAN or empty values in the dataset
data = df.dropna()
data.reset_index(drop=True, inplace=True) # reset index/counter
data.drop_duplicates(subset='Sentence', inplace=True) # drop duplicate values
data['Label'] = panda.to_numeric(data['Label']) # convert to numeric
data.sample(n=5)

del df1, df2, frames
```

[ ]:
```
# get more info about dataset which has been clean
rd_count = df[df.columns[0]].count() # real dataset
cd_count = data[data.columns[0]].count() # cleaned dataset

print("Total of original rows\t:", rd_count)
print("Total of eligible rows\t:", cd_count)
print("Total of removed rows\t:", rd_count - cd_count)
```

```
Total of original rows  : 49823
Total of eligible rows  : 49226
Total of removed rows   : 597
```

[ ]:
```
# remove previous dataset for avoid error
if os.path.exists(corpus_dir):
    shutil.rmtree(corpus_dir)
```

```
pbar  = tqdm_notebook(total=len(data.values), desc="Convert CSV to corpus file")
index = 1
for payload, label in data.values:
    label     = 'payload' if label == 1 else 'non-payload'
    label_dir = os.path.join(corpus_dir, label)
    if not os.path.exists(label_dir):
        os.makedirs(label_dir)
    dt_filepath = os.path.join(label_dir, f'{index}.txt')
    with open(dt_filepath, 'w') as writer:
        writer.write(payload)
    pbar.update(1)
    index += 1
pbar.close()


del df, data
```

```
Convert CSV to corpus file:   0%|          | 0/49226 [00:00<?, ?it/s]
```

Core Steps of Model or Classifier Building

[ ]:
```
from nltk.corpus.reader import CategorizedPlaintextCorpusReader

corpus_set = CategorizedPlaintextCorpusReader(
    corpus_dir, r'(?!\.).*\.txt',
    cat_pattern=r'(payload|non-payload)/.*',
    encoding="utf-8"
)

print('Payload Total\t\t:',   len(corpus_set.fileids('payload')))
print('Non-Payload Total\t:', len(corpus_set.fileids('non-payload')))
print('Total Corpus\t\t:', len(corpus_set.fileids('payload')) + len(corpus_set.
 ↪fileids('non-payload')))
```

```
Payload Total           : 21158
Non-Payload Total       : 28068
Total Corpus            : 49226
```

Corpus Cleaning and Positioning

[ ]:
```
import string
import re, inspect


corpus_rules = {
    'to_lowercase'  : True,
    'only_alphanum' : False,
    'remove_puncts' : False,
```

```
}


def construct_corpus(corpus_in)->list:
    punctuations = list(string.punctuation)
    corpus_out   = []
    if type(corpus_in) == str:
        if corpus_rules['to_lowercase']:
            corpus_in = corpus_in.lower()
        corpus_in = word_tokenize(corpus_in)
    else:
        if corpus_rules['to_lowercase']:
            corpus_in = [w.lower() for w in corpus_in]
    for item in corpus_in:
        if not corpus_rules['remove_puncts']:
            punctuations = []
        if not item in punctuations:
            if corpus_rules['only_alphanum']:
                item = re.sub(r'[\W\d]', '', item)
            if item:
                corpus_out.append(item)
    return corpus_out


def construct_features(corpus:string)->dict:
    feature = {}
    corpus  = set(construct_corpus(corpus))
    for vector in feature_vectors.keys():
        feature[vector] = vector in corpus # corpus string -> corpus list
    return feature


def check_payload(classifier, payload:string):
    predict = classifier.classify(construct_features(payload))
    return predict


def get_accuracy(classifier, test_fs):
    predict = classifier.classify_many([fs for (fs, l) in test_fs])
    correct = [l == r for ((fs, l), r) in zip(test_fs, predict)]
    if correct:
        accuracy = sum(correct) / len(correct)
    else:
        accuracy = 0
    return [predict, accuracy]
```

```
def wordcloud_draw(data, filename, bgcolor='white'):
    words  = ' '.join(data)
    wCloud = WordCloud(background_color=bgcolor, width=2250, height=1000).
 ↪generate(words)
    wCloud.to_file(filename)
    plot.figure(1, figsize=(20, 10))
    plot.imshow(wCloud)
    plot.axis('off')
    plot.show()


def wordplot_draw(freq_words, filename, graph_limit=30):
    fig = plot.figure(figsize=(16, 7))
    plot.gcf().subplots_adjust(bottom=0.15)
    freq_words.plot(graph_limit, title=f'{graph_limit} Most Common Words')
    plot.show()
    plot.draw()
    fig.savefig(filename, dpi=120)


def save_report(classifiers, report_name):
    filename = os.path.join(reporter_dir, f"{report_name}.json")
    reports  = []
    for id, attr in classifiers.items():
        item = {k:str(v) for k,v in attr.items()}
        reports.append(item)
    if not reports:
        return None
    with open(filename, 'w') as writer:
        json.dump(reports, writer)


def get_report_history():
    reports = []
    for report_file in os.listdir(reporter_dir):
        filename = os.path.join(reporter_dir, report_file)
        if filename.endswith('.json'):
            with open(filename) as reader:
                report   = json.load(reader)
                reports += report
    return reports


def show_report_history(n=10):
    reports = get_report_history()
    if reports:
```

```
        tf = panda.DataFrame.from_records([item.values() for item in reports],␣
 ↪columns=[list(reports[0].keys())])
        filepath  = os.path.join(os.getcwd(), report_dir, 'report')
        tf.to_excel(f"{filepath}.xlsx")
        tf.to_html(f"{filepath}.html")
        display(tf.head(n))
```

[ ]:
```
pbar = tqdm_notebook(corpus_set.fileids(), desc="Calculating words count")
words_counter = []
for file_id in corpus_set.fileids():
    word_count = len(corpus_set.words(fileids=file_id))
    words_counter.append(word_count)
    pbar.update(1)
pbar.close()

margin       = 5
vector_limit = int((sum(words_counter)/len(words_counter)) * margin)
print(f'Vector limit is {vector_limit}')
```

```
Calculating words count:   0%|          | 0/49226 [00:00<?, ?it/s]

Vector limit is 505
```

[ ]:
```
feature_vectors = {}
_freq_words     = {} # for informational purpose

for category in corpus_set.categories():
    print(f"Injecting {vector_limit} {category} vectors to feature vectors...")
    corpus_cat = corpus_set.words(categories=[category])
    freq_words = nltk.FreqDist(construct_corpus(corpus_cat))
    most_words = {word.lower():n for word, n in freq_words.
 ↪most_common(vector_limit)}
    _freq_words[category] = [most_words, freq_words]
    feature_vectors.update(most_words)

    t = {f"{word.lower()},{n}" for word, n in freq_words.most_common(50)}
    display(t)


# save feature fectors
filename = f"feature_vector-tfs({len(corpus_set.
 ↪fileids())})-vl({vector_limit})-m({margin})"
cr_name  = '-'.join([f"{k}({int(v)})" for k,v in corpus_rules.items()])
with open(os.path.join(classifier_dir, f'{filename}-{cr_name}.json'), "w") as␣
 ↪writer:
    json.dump(feature_vectors, writer)
```

```
    print('Feature vector saved...')
```

```
Injecting 505 non-payload vectors to feature vectors…
{'%,17570',
 '&,87249',
 "',14665",
 '(,62535',
 '),21407',
 '*,12832',
 '+,32057',
 ',,84845',
 '-,68943',
 '.,219576',
 '/,34175',
 '//,11934',
 ':,36622',
 '://,14920',
 ';,14197',
 '=,130807',
 '\\",14300',
 "\\',19612",
 '\\,97530',
 'a,51306',
 'and,50519',
 'at,14020',
 'but,11056',
 'for,20950',
 'function,16081',
 'hotel,15756',
 'http,14801',
 'i,28392',
 'if,14444',
 'in,26748',
 'is,19495',
 'it,18520',
 'not,10902',
 'of,23893',
 'on,14270',
 'r,18118',
 'room,12663',
 't,115031',
 'that,12647',
 'the,105107',
 'this,29905',
 'to,43662',
 'var,11047',
 'was,29028',
```

```
 'we,25267',
 'were,12462',
 'with,12314',
 'you,12169',
 '{,22878',
 '},12649'}
```

```
Injecting 505 payload vectors to feature vectors…
```

```
{'%,99278',
 '&,35125',
 '(,15584',
 '+,3307',
 ',,10503',
 '-,7237',
 '.,58055',
 '/,34784',
 '1,11427',
 '20,3680',
 '22,8632',
 '29,3293',
 '31,3311',
 '3c,11103',
 '3cscript,6000',
 '3e,21640',
 '3ealert,4976',
 ':,4001',
 '://,15715',
 ';&,16124',
 ';,40825',
 ';/,8273',
 '<,16300',
 '</,4163',
 '=",7074',
 '=%,5953',
 '=&,7013',
 '=,26954',
 '>,18182',
 '?,12513',
 'alert,13801',
 'amp,16526',
 'asp,2903',
 'br,12427',
 'com,8635',
 'cookie,3519',
 'document,3401',
 'gt,23021',
 'h1,3633',
 'http,15110',
```

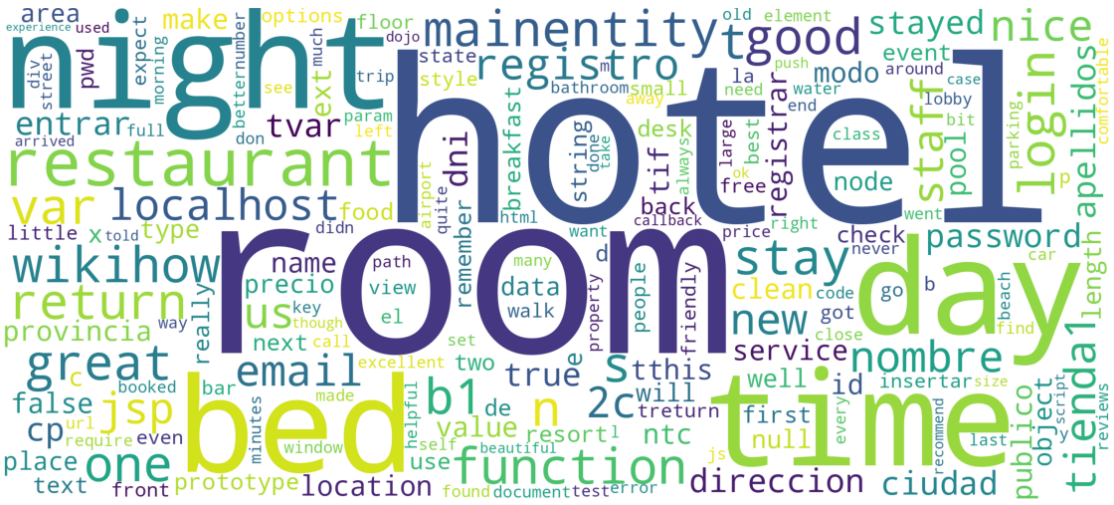
```
 'id,3365',
 'lt,16954',
 'php,5556',
 'quot,8862',
 'script,16217',
 'search,3256',
 'style,3570',
 'test,3579',
 'www,8868',
 'x,3983'}
```

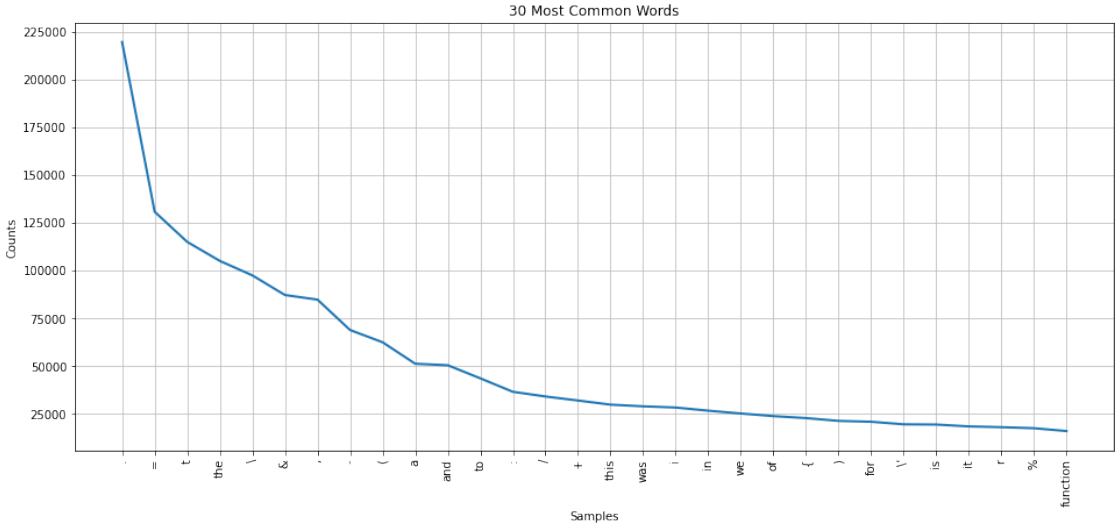
```
Feature vector saved…
```

Informational Purpose

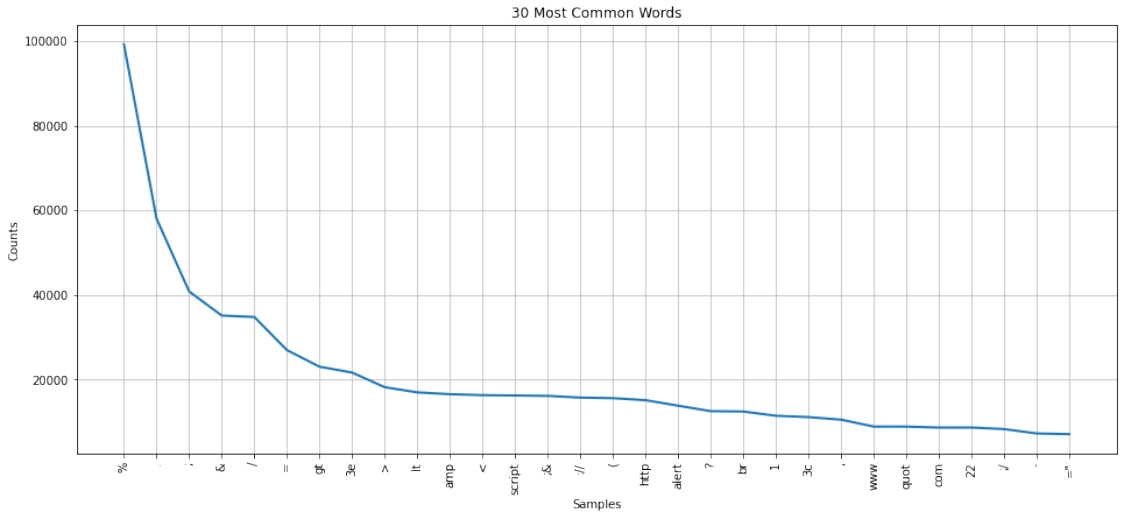
[ ]:
```
graph_limit = 30
figure_dir  = os.path.join(report_dir, 'figures')
if not os.path.exists(figure_dir):
    os.makedirs(figure_dir)

for category, attrs in _freq_words.items():
    print(f'Showing {vector_limit} most common words in {category}...')
    most_word, freq_words = attrs
    filename = f"{category}-tfs({len(corpus_set.
 ↪fileids())})-vl({vector_limit})-m({margin})"
    cr_name  = '-'.join([f"{k}({int(v)})" for k,v in corpus_rules.items()])
    wordcloud_draw(most_word, os.path.join(figure_dir,␣
 ↪f'WC-{filename}-{cr_name}.png'))
    wordplot_draw(freq_words, os.path.join(figure_dir,␣
 ↪f'WP-{filename}-{cr_name}.png'), graph_limit)

del _freq_words
```

```
Showing 505 most common words in non-payload…
```





Showing 505 most common words in payload…



<Figure size 432x288 with 0 Axes>

Preparing Feature Enginering Functions

[ ]:
```
feature_set_dir = os.path.join('output', 'feature-sets')

# remove previous feature_set_dir for avoid error
if os.path.exists(feature_set_dir):
    print('Removing old feature-sets...')
    shutil.rmtree(feature_set_dir)

if not os.path.exists(feature_set_dir):
    os.makedirs(feature_set_dir)


datasets  = []
pbar      = tqdm_notebook(corpus_set.fileids())
step      = 1
for category in corpus_set.categories():
    pbar.set_description(f"({step}/{len(corpus_set.categories())}) Classifying␣
 ↪Courpus -> {category}")
    labels = corpus_set.fileids(category)
    for file_id in labels:
        content = corpus_set.raw(file_id)
        datasets.append((file_id, content, category))
        pbar.update(1)
    step += 1
pbar.close()

del corpus_set
```

```
Removing old feature-sets…

  0%|          | 0/49226 [00:00<?, ?it/s]
```

[ ]:
```
feature_sets = []
pbar         = tqdm_notebook(total=len(datasets), desc=f"Processing Feature Set␣
 ↪with {vector_limit} Vector Limit")
for (file_id, payloads, category) in datasets:
    basename = os.path.basename(file_id)
    filename = os.path.join(feature_set_dir, basename)
    features = construct_features(payloads)
    # with open(filename, 'w') as writer: json.dump([features, category],␣
 ↪writer)
    feature_sets.append([features, category])
    pbar.update(1)
pbar.close()

del datasets
```

```
Processing Feature Set with 505 Vector Limit:   0%|          | 0/49226 [00:00<?,␣
 ↪?it/s]
```

Spliting Data Training and Testing

[ ]:
```
from sklearn import model_selection


test_size = 0.2 # percentation of data testing
X, y      = [x for x,y in feature_sets], [y for x,y in feature_sets]

# split data training and data testing using Python NLTK
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y,␣
 ↪test_size=test_size)

# melihat jumlah data training dan testing
total_train_set     = len(X_train)
total_test_set      = len(X_test)
total_feature_sets  = len(feature_sets)

print('Data Training\t:', total_train_set)
print('Data Testing \t:', total_test_set)

del feature_sets
```

```
Data Training   : 39380
Data Testing    : 9846
```

[ ]:
```
from nltk import classify

scikitNLTK          = classify.scikitlearn.SklearnClassifier
classifiers         = {
    'naive-bayes'   : {
        'name'      : 'Naive Bayes',
        'trainer'   : classify.NaiveBayesClassifier.train,
    },
    'logistic-regress': {
       'name'       : 'Logistic Regression',
       'trainer'    : scikitNLTK(LogisticRegression()).train,
    },
    'gradient-boost': {
       'name'       : 'Gradient Boosting',
       'trainer'    : scikitNLTK(GradientBoostingClassifier()).train,
    },
    'sv-machine': {
        'name'      : 'Support Vector Machine',
        'trainer'   : scikitNLTK(SVC()).train,
    },
    'kn-neighbors'  : {
       'name'       : 'K-Nearest Neigbors',
       'trainer'    : scikitNLTK(KNeighborsClassifier(n_neighbors=3)).train,
    },
}
```

[ ]:
```
print(f"Training information -> vector_limit {vector_limit} | margin: {margin}␣
 ↪| test_size: {test_size}...")
train_featureset    = [[X_train[i], y_train[i]] for i in range(total_train_set)]
test_featureset     = [[X_test[i], y_test[i]] for i in range(total_test_set)]

counter = 1
for id, attr in classifiers.items():
    print(f"[{counter}] Training data using {attr['name']}...")
    start_time          = datetime.now()
    attr['classifier']  = attr['trainer'](train_featureset)
    attr['y_predict'], attr['accuracy'] = get_accuracy(attr['classifier'],␣
 ↪test_featureset)
    attr['elapse_time'] = str(datetime.now() - start_time)
    print(f"Accuracy Level\t: {attr['accuracy']}")
    print(f"Time Ellapsed\t: {attr['elapse_time']}\n")
    counter += 1


print(f'Ordering classifier based on accuracy level...')
index       = 1
```

```
classifiers  = {k: v for k, v in sorted(classifiers.items(), key=lambda item:␣
 ↪item[1]['accuracy'], reverse=True)}
for id, attr in classifiers.items():
    print(f"[{index}] {attr['name']} (Accuracy: {attr['accuracy']}) (Time␣
 ↪Ellapse: {attr['elapse_time']})")
    index += 1

del train_featureset, test_featureset
```

```
Training information -> vector_limit 505 | margin: 5 | test_size: 0.2…
[1] Training data using Naive Bayes…
0.7032297379646557
Accuracy Level  : 0.7032297379646557
Time Ellapsed   : 0:00:44.060281

[2] Training data using Logistic Regression…
0.9947186674791794
Accuracy Level  : 0.9947186674791794
Time Ellapsed   : 0:00:40.664987

[3] Training data using Gradient Boosting…
0.9922811293926468
Accuracy Level  : 0.9922811293926468
Time Ellapsed   : 0:01:53.887530

[4] Training data using Support Vector Machine…
0.9965468210440788
Accuracy Level  : 0.9965468210440788
Time Ellapsed   : 0:03:46.050874

[5] Training data using K-Nearest Neigbors…
0.9958358724355068
Accuracy Level  : 0.9958358724355068
Time Ellapsed   : 0:21:38.650214

Ordering classifier based on accuracy level…
[1] Support Vector Machine (Accuracy: 0.9965468210440788) (Time Ellapse:
0:03:46.050874)
[2] K-Nearest Neigbors (Accuracy: 0.9958358724355068) (Time Ellapse:
0:21:38.650214)
[3] Logistic Regression (Accuracy: 0.9947186674791794) (Time Ellapse:
0:00:40.664987)
[4] Gradient Boosting (Accuracy: 0.9922811293926468) (Time Ellapse:
0:01:53.887530)
[5] Naive Bayes (Accuracy: 0.7032297379646557) (Time Ellapse: 0:00:44.060281)
```

Classifier Testing with Small Sampling

[ ]:
```
custom_payloads = [
    ("Teks html biasa <h1>judul</h1>", 0),
    ("<h1>Teks biasa</h1>", 0),
    ("<div>Bukan kode serangan</div>", 0),
    ("<SCRIPT>prompt(1)</script>", 1),
    ("<SCRIPT>document.URL; alert(1)", 0),
    ("<h1>ini adalah teks HTML biasa script script</h1>", 0),
]
```

[ ]:
```
for id, attr in classifiers.items():
    table_row, tracks = [], []
    for payload, label in custom_payloads:
        label      = 'payload' if label == 1 else 'non-payload'
        prediction = check_payload(attr['classifier'], payload)
        status     = label == prediction
        table_row.append([payload, label, prediction, status])
        tracks.append(status)
    table_frame = panda.DataFrame.from_records(table_row, columns=['Payload',␣
 ↪'Actual', 'Predict', 'Status'])
    valid_preds = []
    for status in tracks:
        if status is True: valid_preds.append(status)
    display_html(f"<h3>{attr['name']} Prediction Result ({len(valid_preds)}/
 ↪{len(tracks)})</h3><hr/>", raw=True)
    display(table_frame)
```

```
                                        Payload       Actual  \
0                    Teks html biasa <h1>judul</h1>  non-payload
1                              <h1>Teks biasa</h1>  non-payload
2                    <div>Bukan kode serangan</div>  non-payload
3                        <SCRIPT>prompt(1)</script>      payload
4                    <SCRIPT>document.URL; alert(1)  non-payload
5  <h1>ini adalah teks HTML biasa script script</h1>  non-payload

       Predict  Status
0      payload   False
1      payload   False
2      payload   False
3      payload    True
4      payload   False
5  non-payload    True
```

```
                                        Payload       Actual  \
0                    Teks html biasa <h1>judul</h1>  non-payload
1                              <h1>Teks biasa</h1>  non-payload
2                    <div>Bukan kode serangan</div>  non-payload
3                        <SCRIPT>prompt(1)</script>      payload
4                    <SCRIPT>document.URL; alert(1)  non-payload
```

```
5  <h1>ini adalah teks HTML biasa script script</h1>  non-payload

       Predict  Status
0      payload   False
1      payload   False
2      payload   False
3      payload    True
4      payload   False
5  non-payload    True

                                          Payload       Actual  \
0                    Teks html biasa <h1>judul</h1>  non-payload
1                              <h1>Teks biasa</h1>  non-payload
2                    <div>Bukan kode serangan</div>  non-payload
3                        <SCRIPT>prompt(1)</script>      payload
4                    <SCRIPT>document.URL; alert(1)  non-payload
5  <h1>ini adalah teks HTML biasa script script</h1>  non-payload

       Predict  Status
0      payload   False
1      payload   False
2      payload   False
3      payload    True
4      payload   False
5  non-payload    True

                                          Payload       Actual  \
0                    Teks html biasa <h1>judul</h1>  non-payload
1                              <h1>Teks biasa</h1>  non-payload
2                    <div>Bukan kode serangan</div>  non-payload
3                        <SCRIPT>prompt(1)</script>      payload
4                    <SCRIPT>document.URL; alert(1)  non-payload
5  <h1>ini adalah teks HTML biasa script script</h1>  non-payload

       Predict  Status
0      payload   False
1      payload   False
2      payload   False
3      payload    True
4      payload   False
5  non-payload    True

                                          Payload       Actual  Predict  \
0                    Teks html biasa <h1>judul</h1>  non-payload  payload
1                              <h1>Teks biasa</h1>  non-payload  payload
2                    <div>Bukan kode serangan</div>  non-payload  payload
3                        <SCRIPT>prompt(1)</script>      payload  payload
4                    <SCRIPT>document.URL; alert(1)  non-payload  payload
5  <h1>ini adalah teks HTML biasa script script</h1>  non-payload  payload
```

```
   Status
0   False
1   False
2   False
3    True
4   False
5   False
```

[ ]:
```
df_challenge          = panda.read_csv(challenge_csv)
df_challenge['Label'] = panda.to_numeric(df_challenge['Label'])
df_challenge.drop(['Unnamed: 0'], axis=1, inplace=True)
df_challenge.drop_duplicates(inplace=True) # drop duplicate
df_challenge.dropna(subset='Sentence', inplace=True) # drop empty values
display_html(f"<h3>Total data will be predicted by classifier:␣
 ↪{len(df_challenge.values)}</h3><hr/>", raw=True)

df_challenge.sample(n=5)
```

[ ]:
```
                                                  Sentence  Label
9140  <frameset draggable="true" ondragenter="alert(…      1
4616  \t </span><a href="/w/index.php?title=Artifici…      0
7954           <hgroup onclick="alert(1)">test</hgroup>      1
7327  <style>@keyframes x{from {left:0;}to {left: 10…      1
4173                 <ol onpointerout=alert(1)>XSS</ol>      1
```

[ ]:
```
step = 1
pbar = tqdm_notebook()
for id, attr in classifiers.items():
    pbar.reset(total=len(df_challenge.values))
    pbar.set_description_str(f"({step}/{len(classifiers)}) Predicting with␣
 ↪{attr['name']}")
    tracks     = []
    start_time = datetime.now()
    for payload, label in df_challenge.values:
        label      = 'payload' if label == 1 else 'non-payload'
        prediction = check_payload(attr['classifier'], payload)
        status     = label == prediction
        tracks.append(status)
        pbar.update(1)
    valid_preds = []
    for status in tracks:
        if status is True: valid_preds.append(status)

    ellapse_time        = str(datetime.now() - start_time)
    attr['challenges']  = {
        'total_data'    : len(tracks),
```

```
        'total_valid'   : len(valid_preds),
        'total_invalid' : (len(tracks) - len(valid_preds)),
        'ellapse_time'  : ellapse_time,
    }
    percentation = int((len(valid_preds)/len(tracks)) * 100)
    information  = ' | '.join([f"{k.capitalize()}: {v}" for k,v in␣
 ↪attr['challenges'].items()])
    display_html(f"<div>{attr['name']} Prediction ({percentation}%)</div>",␣
 ↪raw=True)
    display_html(f"<div>{information}<hr/></div>", raw=True)
    step += 1
pbar.close()

total_payload_challenge = len(df_challenge)
del df_challenge
```

```
0it [00:00, ?it/s]
```

[ ]:
```
def get_confussion_matrix(TP, FP, FN):
    confussion_matrix = {}
    for label in sorted(labels):
        precision, recall = 0, 0
        if TP[label] == 0:
            f_measure = 0
        else:
            precision = float(TP[label]) / (TP[label] + FP[label])
            recall    = float(TP[label]) / (TP[label] + FN[label])
            f_measure = float(2) * (precision * recall) / (precision + recall)
        confussion_matrix[label] = {
            'label'     : label,
            'precision' : precision,
            'recall'    : recall,
            'f_measure' : f_measure,
        }
    return confussion_matrix


def table_confussion_matrix(confussion_matrix):
    table_rows         = [v.values() for k,v in confussion_matrix.items()]
    table_frame        = panda.DataFrame.from_records(table_rows,␣
 ↪columns=['Label', 'Precision', 'Recall', 'F-Measure'])
    return table_frame
```

[ ]:
```
from collections import Counter

step = 1
for id, attr in classifiers.items():
```

```
    nltk_cm = nltk.ConfusionMatrix(y_test, attr['y_predict'], True)
    labels  = {'payload', 'non-payload'}

    TP, FN, FP = Counter(), Counter(), Counter()
    for i in labels:
        for j in labels:
            if i == j:
                TP[i] += int(nltk_cm[i,j])
            else:
                FN[i] += int(nltk_cm[i,j])
                FP[j] += int(nltk_cm[i,j])

    confussion_matrix         = get_confussion_matrix(TP, FP, FN)
    attr['confussion_matrix'] = confussion_matrix
    print(f"Classifier Name {attr['name']}")
    print(nltk_cm.pretty_format())
    display(table_confussion_matrix(confussion_matrix))
    print(f"With accuracy level -> {attr['accuracy']}")
    display_html('<hr/>', raw=True)
    step += 1

    # remove unused variable
    del attr['y_predict']
```

```
Classifier Name Support Vector Machine
             |    n      |
             |    o      |
             |    n      |
             |    -      |
             |    p    p |
             |    a    a |
             |    y    y |
             |    l    l |
             |    o    o |
             |    a    a |
             |    d    d |
-------------+-----------+
non-payload |<5573>  15 |
    payload |   19<4239>|
-------------+-----------+
(row = reference; col = test)
```

	Label	Precision	Recall	F-Measure
0	non-payload	0.996602	0.997316	0.996959
1	payload	0.996474	0.995538	0.996006

```
With accuracy level -> 0.9965468210440788
```

```
Classifier Name K-Nearest Neigbors
            |    n      |
            |    o      |
            |    n      |
            |    -      |
            |    p    p |
            |    a    a |
            |    y    y |
            |    l    l |
            |    o    o |
            |    a    a |
            |    d    d |
------------+-----------+
non-payload |<5570>  18 |
    payload |   23<4235>|
------------+-----------+
(row = reference; col = test)


         Label  Precision    Recall  F-Measure
0  non-payload   0.995888  0.996779   0.996333
1      payload   0.995768  0.994598   0.995183

With accuracy level -> 0.9958358724355068

Classifier Name Logistic Regression
            |    n      |
            |    o      |
            |    n      |
            |    -      |
            |    p    p |
            |    a    a |
            |    y    y |
            |    l    l |
            |    o    o |
            |    a    a |
            |    d    d |
------------+-----------+
non-payload |<5566>  22 |
    payload |   30<4228>|
------------+-----------+
(row = reference; col = test)


         Label  Precision    Recall  F-Measure
0  non-payload   0.994639  0.996063   0.995351
1      payload   0.994824  0.992954   0.993888

With accuracy level -> 0.9947186674791794

Classifier Name Gradient Boosting
```

```
             |    n      |
             |    o      |
             |    n      |
             |    -      |
             |    p    p |
             |    a    a |
             |    y    y |
             |    l    l |
             |    o    o |
             |    a    a |
             |    d    d |
-------------+-----------+
non-payload |<5558>  30 |
    payload |   46<4212>|
-------------+-----------+
(row = reference; col = test)
```

	Label	Precision	Recall	F-Measure
0	non-payload	0.991792	0.994631	0.993209
1	payload	0.992928	0.989197	0.991059

With accuracy level -> 0.9922811293926468

```
Classifier Name Naive Bayes
             |         n |
             |         o |
             |         n |
             |         - |
             |    p    p |
             |    a    a |
             |    y    y |
             |    l    l |
             |    o    o |
             |    a    a |
             |    d    d |
-------------+-----------+
    payload |<4252>   6 |
non-payload | 2916<2672>|
-------------+-----------+
(row = reference; col = test)
```

	Label	Precision	Recall	F-Measure
0	non-payload	0.997760	0.478168	0.646504
1	payload	0.593192	0.998591	0.744267

With accuracy level -> 0.7032297379646557

[ ]:
```
print('Saving Accuracy Level into file...')

for id, attr in classifiers.items():
    attr['datetime']         = datetime.now()
    attr['margin']           = margin
    attr['test_size']        = test_size
    attr['vector_limit']     = vector_limit
    attr['total_test']       = total_test_set
    attr['total_train']      = total_train_set
    attr['total_featureset'] = total_feature_sets
    attr.update(corpus_rules) # update taken from corpus rules


model_basename =␣
 ↪f"tfs({total_feature_sets})-vl({vector_limit})-m({margin})-{cr_name}"
save_report(classifiers, f"{model_basename}")

display_html('<h3>Display Accuracy History</h3><hr/>')
show_report_history(10)
```

```
Saving Accuracy Level into file…
```

	name	trainer	\
0	Support Vector Machine	<bound method SklearnClassifier.train of <Skle…	
1	Logistic Regression	<bound method SklearnClassifier.train of <Skle…	
2	K-Nearest Neigbors	<bound method SklearnClassifier.train of <Skle…	
3	Gradient Boosting	<bound method SklearnClassifier.train of <Skle…	
4	Naive Bayes	<bound method NaiveBayesClassifier.train of <c…	
5	K-Nearest Neigbors	<bound method SklearnClassifier.train of <Skle…	
6	Support Vector Machine	<bound method SklearnClassifier.train of <Skle…	
7	Gradient Boosting	<bound method SklearnClassifier.train of <Skle…	
8	Logistic Regression	<bound method SklearnClassifier.train of <Skle…	
9	Naive Bayes	<bound method NaiveBayesClassifier.train of <c…	

	classifier	accuracy	\
0	<SklearnClassifier(SVC())>	0.9902498476538696	
1	<SklearnClassifier(LogisticRegression())>	0.9899451553930531	
2	<SklearnClassifier(KNeighborsClassifier(n_neig…	0.9887263863497867	
3	<SklearnClassifier(GradientBoostingClassifier())>	0.9796871826122283	
4	<nltk.classify.naivebayes.NaiveBayesClassifier…	0.6616900264066626	
5	<SklearnClassifier(KNeighborsClassifier(n_neig…	0.9959374365224457	
6	<SklearnClassifier(SVC())>	0.9958358724355068	
7	<SklearnClassifier(GradientBoostingClassifier())>	0.9929920780012188	
8	<SklearnClassifier(LogisticRegression())>	0.992788949827341	
9	<nltk.classify.naivebayes.NaiveBayesClassifier…	0.7200893763965062	

	elapse_time	challenges	\
0	0:05:04.857040	{'total_data': 10917, 'total_valid': 8499, 'to…	

```
1  0:00:38.568305  {'total_data': 10917, 'total_valid': 9167, 'to…
2  0:14:23.166549  {'total_data': 10917, 'total_valid': 7816, 'to…
3  0:01:51.525164  {'total_data': 10917, 'total_valid': 8962, 'to…
4  0:00:39.536707  {'total_data': 10917, 'total_valid': 7927, 'to…
5  0:05:47.721164  {'total_data': 10917, 'total_valid': 7512, 'to…
6  0:01:20.071821  {'total_data': 10917, 'total_valid': 7688, 'to…
7  0:00:53.146987  {'total_data': 10917, 'total_valid': 7628, 'to…
8  0:00:16.435809  {'total_data': 10917, 'total_valid': 7939, 'to…
9  0:00:15.377279  {'total_data': 10917, 'total_valid': 8100, 'to…

                                  confussion_matrix  \
0  {'non-payload': {'label': 'non-payload', 'prec…
1  {'non-payload': {'label': 'non-payload', 'prec…
2  {'non-payload': {'label': 'non-payload', 'prec…
3  {'non-payload': {'label': 'non-payload', 'prec…
4  {'non-payload': {'label': 'non-payload', 'prec…
5  {'non-payload': {'label': 'non-payload', 'prec…
6  {'non-payload': {'label': 'non-payload', 'prec…
7  {'non-payload': {'label': 'non-payload', 'prec…
8  {'non-payload': {'label': 'non-payload', 'prec…
9  {'non-payload': {'label': 'non-payload', 'prec…

                    datetime margin test_size vector_limit total_test  \
0  2022-06-19 22:13:24.867594      7       0.2          707       9846
1  2022-06-19 22:13:24.867605      7       0.2          707       9846
2  2022-06-19 22:13:24.867607      7       0.2          707       9846
3  2022-06-19 22:13:24.867609      7       0.2          707       9846
4  2022-06-19 22:13:24.867610      7       0.2          707       9846
5  2022-06-20 16:08:57.950639    2.5       0.2          252       9846
6  2022-06-20 16:08:57.950658    2.5       0.2          252       9846
7  2022-06-20 16:08:57.950662    2.5       0.2          252       9846
8  2022-06-20 16:08:57.950664    2.5       0.2          252       9846
9  2022-06-20 16:08:57.950665    2.5       0.2          252       9846

  total_train total_featureset to_lowercase only_alphanum remove_puncts
0       39380            49226         True          True          True
1       39380            49226         True          True          True
2       39380            49226         True          True          True
3       39380            49226         True          True          True
4       39380            49226         True          True          True
5       39380            49226         True         False         False
6       39380            49226         True         False         False
7       39380            49226         True         False         False
8       39380            49226         True         False         False
9       39380            49226         True         False         False
```

[ ]:
```python
import pickle


save_classifier = True

if save_classifier:
    print("Saving classifier into file/object based...")
    for id, attr in classifiers.items():
        dirname = os.path.join(classifier_dir, model_basename)
        if not os.path.exists(dirname):
            os.makedirs(dirname)
        filename = os.path.join(dirname, f"{id}.model")
        # save models
        with open(filename, "wb") as writer:
            pickle.dump(attr['classifier'], writer)


def load_classifier(name):
    filename = os.path.join(classifier_dir, f"{name}.model")
    if not os.path.exists(filename):
        raise ValueError(f"No classifier with name {name}")
    with open(filename, 'rb') as reader:
        classifier = pickle.load(reader)
        return classifier
```

```
Saving classifier into file/object based…
```