

Додаток А

preprocessing.ipynb

```
## CICIDS2017 csv files are required for the operation of the program.
## These files must be located under the "CSVs" folder in the same directory
as the program.

## The purpose of this program is to clear the csv files containing
CICIDS2017 data from errors.
## the faults observed are:
##      1- 288602 of the entries in the file "Thursday-WorkingHours-
Morning-WebAttacks.pcap_ISCX.csv" are empty / meaningless.
##      (e.g.
"/,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
,,,,,,,,")
##
##      2- In the original csv files, while describing the Web Attack types
such as Brute Force, XSS, Sql Injection, the character used is not recognized
##      by the Python-Pandas library and leads to the error.
##      this character ("-", Unicode code:8211) has been
changed with another character ("_", Unicode code:45) to correct the error.
##
## After the error correction, all the csv files were made into a single
file (all_date.csv) to make it easier to process.

import pandas as pd
import os
from sklearn import preprocessing
import time
seconds = time.time()
%matplotlib inline

print("This process may take 5 to 10 minutes, depending on the performance of
your computer.\n\n\n")
number="0123456789"
# CSV files names:
csv_files=["Monday-WorkingHours.pcap_ISCX",
           "Tuesday-WorkingHours.pcap_ISCX",
           "Wednesday-workingHours.pcap_ISCX",
           "Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX",
           "Thursday-WorkingHours-Afternoon-Infiltration.pcap_ISCX",
           "Friday-WorkingHours-Morning.pcap_ISCX",
           "Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX",
           "Friday-WorkingHours-Afternoon-DDos.pcap_ISCX",]

# Headers of column
main_labels=["Flow ID","Source IP","Source Port","Destination
IP","Destination Port","Protocol","Timestamp","Flow Duration","Total Fwd
Packets",
```

```

    "Total Backward Packets","Total Length of Fwd Packets","Total Length of
    Bwd Packets","Fwd Packet Length Max","Fwd Packet Length Min",
    "Fwd Packet Length Mean","Fwd Packet Length Std","Bwd Packet Length
    Max","Bwd Packet Length Min","Bwd Packet Length Mean","Bwd Packet Length
    Std",
    "Flow Bytes/s","Flow Packets/s","Flow IAT Mean","Flow IAT Std","Flow IAT
    Max","Flow IAT Min","Fwd IAT Total","Fwd IAT Mean","Fwd IAT Std","Fwd IAT
    Max",
    "Fwd IAT Min","Bwd IAT Total","Bwd IAT Mean","Bwd IAT Std","Bwd IAT
    Max","Bwd IAT Min","Fwd PSH Flags","Bwd PSH Flags","Fwd URG Flags","Bwd URG
    Flags",
    "Fwd Header Length","Bwd Header Length","Fwd Packets/s","Bwd
    Packets/s","Min Packet Length","Max Packet Length","Packet Length
    Mean","Packet Length Std",
    "Packet Length Variance","FIN Flag Count","SYN Flag Count","RST Flag
    Count","PSH Flag Count","ACK Flag Count","URG Flag Count","CWE Flag Count",
    "ECE Flag Count","Down/Up Ratio","Average Packet Size","Avg Fwd Segment
    Size","Avg Bwd Segment Size","faulty-Fwd Header Length","Fwd Avg Bytes/Bulk",
    "Fwd Avg Packets/Bulk","Fwd Avg Bulk Rate","Bwd Avg Bytes/Bulk","Bwd Avg
    Packets/Bulk","Bwd Avg Bulk Rate","Subflow Fwd Packets","Subflow Fwd Bytes",
    "Subflow Bwd Packets","Subflow Bwd
    Bytes","Init_Win_bytes_forward","Init_Win_bytes_backward","act_data_pkt_fwd",
    "min_seg_size_forward","Active Mean","Active Std","Active Max","Active
    Min","Idle Mean","Idle Std","Idle Max","Idle Min","Label","External IP"]

main_labels2=main_labels
main_labels=( ",".join( i for i in main_labels ) )
main_labels=main_labels+"\n"
flag=True
for i in range(len(csv_files)):
    ths = open(str(i)+".csv", "w")
    ths.write(main_labels)
    with open("./CSVs/"+csv_files[i]+".csv", "r") as file:
        while True:
            try:
                line=file.readline()
                if line[0] in number:# this line eliminates the headers of
                CSV files and incomplete streams .
                    if " - " in str(line): ## if there is "-" character ("-
                    ", Unicode code:8211) in the flow , it will be chanced with "-" character (
                    Unicode code:45).
                        line=(str(line).replace(" - "," - "))
                        line=(str(line).replace("inf","0"))
                        line=(str(line).replace("Infinity","0"))
                        line=(str(line).replace("NaN","0"))

                        ths.write(str(line))
                    else:
                        continue
            except:
                break
    ths.close()

df=pd.read_csv(str(i)+".csv",low_memory=False)
df=df.fillna(0)

```

```

string_features=["Flow Bytes/s","Flow Packets/s"]
for ii in string_features: #Some data in the "Flow Bytes / s" and "Flow Packets / s" columns are not numeric. Fixing this bug in this loop
    df[ii]=df[ii].replace('Infinity', -1)
    df[ii]=df[ii].replace('NaN', 0)
    number_or_not=[]
    for iii in df[ii]:
        try:
            k=int(float(iii))
            number_or_not.append(int(k))
        except:
            number_or_not.append(iii)
    df[ii]=number_or_not

string_features=[]
for j in main_labels2: # In this section, non-numeric (string and / or categorical) properties (columns) are detected.
    if df[j].dtype=="object":
        string_features.append(j)
try:
    string_features.remove('Label')#The "Label" property was removed from the list. Because it has to remain "categorical" for using with different machine learning approach.
except:
    print("error!")
    labelencoder_X = preprocessing.LabelEncoder()

for ii in string_features: ## In this loop, non-numeric (string and/or categorical) properties converted to numeric features.
    try:
        df[ii]=labelencoder_X.fit_transform(df[ii])
    except:
        df[ii]=df[ii].replace('Infinity', -1)
    df=df.drop(main_labels2[61], axis=1) ## Column 61 is deleted because it is unnecessary, column 41 ("Fwd Header Length" feature) had be mistakenly rewritten.

##All CSV files are merged into a single file.
if flag:
    df.to_csv('all_data.csv' ,index = False)
    flag=False
else:
    df.to_csv('all_data.csv' ,index = False,header=False,mode="a")
    os.remove(str(i)+".csv")
    print("The pre-processing phase of the ",csv_files[i]," file is completed.\n")

print("mission accomplished!")

```

```
print("Total operation time: = ",time.time()- seconds ,"seconds")
```

This process may take 5 to 10 minutes, depending on the performance of your computer.

The pre-processing phase of the Monday-WorkingHours.pcap_ISCX file is completed.

The pre-processing phase of the Tuesday-WorkingHours.pcap_ISCX file is completed.

The pre-processing phase of the Wednesday-workingHours.pcap_ISCX file is completed.

The pre-processing phase of the Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX file is completed.

The pre-processing phase of the Thursday-WorkingHours-Afternoon-Infiltration.pcap_ISCX file is completed.

The pre-processing phase of the Friday-WorkingHours-Morning.pcap_ISCX file is completed.

The pre-processing phase of the Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX file is completed.

The pre-processing phase of the Friday-WorkingHours-Afternoon-DDos.pcap_ISCX file is completed.

mission accomplished!

Total operation time: = 434.0625455379486 seconds

statistics.ipynb

```
## all_data.csv file is required for the operation of the program.
## all_data.csv file must be located in the same directory as the program.
```

```
## The purpose of this program is to produce CSV files consisting of only
one type of attack and benign flow.
## These files contain all attack flow and some benign data flow. The rate :
(attack= 30% , benign=70%)
## normal data streams are randomly selected
```

```
import random
import os
import pandas as pd
import time
```

```

seconds = time.time()
%matplotlib inline

def folder(f_name): #this function creates a folder named "attacks" in the
program directory.
    try:
        if not os.path.exists(f_name):
            os.makedirs(f_name)
    except OSError:
        print ("Tthe folder could not be created!")

print("This process may take 3 to 8 minutes, depending on the performance of
your computer.\n\n\n")

# Headers of column
main_labels=["Flow ID","Source IP","Source Port","Destination
IP","Destination Port","Protocol","Timestamp","Flow Duration","Total Fwd
Packets",
    "Total Backward Packets","Total Length of Fwd Packets","Total Length of
Bwd Packets","Fwd Packet Length Max","Fwd Packet Length Min",
    "Fwd Packet Length Mean","Fwd Packet Length Std","Bwd Packet Length
Max","Bwd Packet Length Min","Bwd Packet Length Mean","Bwd Packet Length
Std",
    "Flow Bytes/s","Flow Packets/s","Flow IAT Mean","Flow IAT Std","Flow IAT
Max","Flow IAT Min","Fwd IAT Total","Fwd IAT Mean","Fwd IAT Std","Fwd IAT
Max",
    "Fwd IAT Min","Bwd IAT Total","Bwd IAT Mean","Bwd IAT Std","Bwd IAT
Max","Bwd IAT Min","Fwd PSH Flags","Bwd PSH Flags","Fwd URG Flags","Bwd URG
Flags",
    "Fwd Header Length","Bwd Header Length","Fwd Packets/s","Bwd
Packets/s","Min Packet Length","Max Packet Length","Packet Length
Mean","Packet Length Std",
    "Packet Length Variance","FIN Flag Count","SYN Flag Count","RST Flag
Count","PSH Flag Count","ACK Flag Count","URG Flag Count","CWE Flag Count",
    "ECE Flag Count","Down/Up Ratio","Average Packet Size","Avg Fwd Segment
Size","Avg Bwd Segment Size","Fwd Avg Bytes/Bulk",
    "Fwd Avg Packets/Bulk","Fwd Avg Bulk Rate","Bwd Avg Bytes/Bulk","Bwd Avg
Packets/Bulk","Bwd Avg Bulk Rate","Subflow Fwd Packets","Subflow Fwd Bytes",
    "Subflow Bwd Packets","Subflow Bwd
Bytes","Init_Win_bytes_forward","Init_Win_bytes_backward","act_data_pkt_fwd",
    "min_seg_size_forward","Active Mean","Active Std","Active Max","Active
Min","Idle Mean","Idle Std","Idle Max","Idle Min","Label","External IP"]
main_labels=( " ".join( i for i in main_labels ) )

attacks=["BENIGN", "Bot", "DDoS", "DoS GoldenEye", "DoS Hulk", "DoS
Slowhttptest", "DoS slowloris", "FTP-Patator", "Heartbleed", "Infiltration",
"PortScan", "SSH-Patator", "Web Attack - Brute Force", "Web Attack - Sql
Injection", "Web Attack - XSS"]
folder("./attacks/")

benign=2359289

```

```
dict_attack={
"Bot":1966,
"DDoS":41835,
"DoS GoldenEye":10293,
"DoS Hulk":231073,
"DoS Slowhttptest":5499,
"DoS slowloris":5796,
"FTP-Patator":7938,
"Heartbleed":11,
"Infiltration":36,
"PortScan":158930,
"SSH-Patator":5897,
"Web Attack - Brute Force":1507,
"Web Attack - XSS":652,
"Web Attack - Sql Injection":21}
```

```
for i in dict_attack: # in this section, a file is opened for each attack
type and is recorded at a random benign flow.
```

```
    a,b=0,0
    ths = open(".\\attacks\\"+i + ".csv", "w")
    ths.write(str(main_labels)+"\n")
    benign_num=int(benign/(dict_attack[i]*(7/3)))
    with open("all_data.csv", "r") as file:
        while True:
            try:
                line=file.readline()
                line=line[:-1]
                k=line.split(",")
                if k[83]=="BENIGN":
                    rnd=random.randint(1,benign_num)
                    if rnd==1:
                        ths.write(str(line)+"\n")
                        b+=1
                if k[83]==i:
                    ths.write(str(line)+"\n")
                    a+=1
                else:
                    continue
            except:
                break

    ths.close()
    print(i , "file is completed\n attack:%d\n benign:%d\n\n\n " %(a,b))
```

```
##All web attack files are merged into a single file.
```

```
webs=["Web Attack - Brute Force","Web Attack - XSS","Web Attack - Sql
Injection"]
flag=True
for i in webs:
    df=pd.read_csv(".\\attacks\\"+str(i)+".csv")
    if flag:
        df.to_csv('.\\attacks\\Web Attack.csv' ,index = False)
    flag=False
```

```

    else:
        df.to_csv('..\\attacks\\Web Attack.csv' ,index =
False,header=False,mode="a")
        os.remove("..\\attacks\\"+str(i)+".csv")

print("mission accomplished!")
print("operation time: = ",time.time()- seconds ,"seconds")
This process may take 3 to 8 minutes, depending on the performance of your co
mputer.

```

```

Bot file is completed
attack:1966
benign:4778

```

```

DDoS file is completed
attack:41835
benign:99398

```

```

DoS GoldenEye file is completed
attack:10293
benign:24105

```

```

DoS Hulk file is completed
attack:231073
benign:591524

```

```

DoS Slowhttptest file is completed
attack:5499
benign:12795

```

```

DoS slowloris file is completed
attack:5796
benign:13296

```

```

FTP-Patator file is completed
attack:7938
benign:18693

```

```

Heartbleed file is completed

```

```
attack:11
benign:26
```

```
Infiltration file is completed
attack:36
benign:94
```

```
PortScan file is completed
attack:158930
benign:393748
```

```
SSH-Patator file is completed
attack:5897
benign:13778
```

```
Web Attack - Brute Force file is completed
attack:1507
benign:3460
```

```
Web Attack - XSS file is completed
attack:652
benign:1594
```

```
Web Attack - Sql Injection file is completed
attack:21
benign:49
```

```
mission accomplished!
operation time: = 303.9238567352295 seconds
```

In []:

feature_selection_for_attack_files.ipynb

```
## "attacks" folder (with attack csv files) is required for the operation of
the program.
## "attacks" folder must be located in the same directory as the program.
```

```
## the purpose of this code is to determine which features to use in the
machine learning phase.
```



```

## for this purpose, the importance weights of the attacks are calculated.
## this calculation was made using sklearn-RandomForestRegressor.

## the some codes parts used for calculation and graphing are taken from the
## following site.
## http://scikit-
## learn.org/stable/auto_examples/ensemble/plot_forest_importances.html

import numpy as np
import os
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import RandomForestRegressor
import sklearn as sk
import time
seconds = time.time()

def folder(f_name): #this function creates a folder named "feature_pics" in
the program directory.
    try:
        if not os.path.exists(f_name):
            os.makedirs(f_name)
    except OSError:
        print ("The folder could not be created!")

# CSV files names:
csv_files=os.listdir("attacks") # It creates a list of file names in the
"attacks" folder.

# Headers of column
main_labels=["Flow Duration","Total Fwd Packets",    "Total Backward
Packets","Total Length of Fwd Packets","Total Length of Bwd Packets","Fwd
Packet Length Max","Fwd Packet Length Min",
    "Fwd Packet Length Mean","Fwd Packet Length Std","Bwd Packet Length
Max","Bwd Packet Length Min","Bwd Packet Length Mean","Bwd Packet Length
Std",
    "Flow Bytes/s","Flow Packets/s","Flow IAT Mean","Flow IAT Std","Flow IAT
Max","Flow IAT Min","Fwd IAT Total","Fwd IAT Mean","Fwd IAT Std","Fwd IAT
Max",
    "Fwd IAT Min","Bwd IAT Total","Bwd IAT Mean","Bwd IAT Std","Bwd IAT
Max","Bwd IAT Min","Fwd PSH Flags","Bwd PSH Flags","Fwd URG Flags","Bwd URG
Flags",
    "Fwd Header Length","Bwd Header Length","Fwd Packets/s","Bwd
Packets/s","Min Packet Length","Max Packet Length","Packet Length
Mean","Packet Length Std",
    "Packet Length Variance","FIN Flag Count","SYN Flag Count","RST Flag
Count","PSH Flag Count","ACK Flag Count","URG Flag Count","CWE Flag Count",
    "ECE Flag Count","Down/Up Ratio","Average Packet Size","Avg Fwd Segment
Size","Avg Bwd Segment Size","Fwd Avg Bytes/Bulk",
    "Fwd Avg Packets/Bulk","Fwd Avg Bulk Rate","Bwd Avg Bytes/Bulk","Bwd Avg
Packets/Bulk","Bwd Avg Bulk Rate","Subflow Fwd Packets","Subflow Fwd Bytes",
    "Subflow Bwd Packets","Subflow Bwd
Bytes","Init_Win_bytes_forward","Init_Win_bytes_backward","act_data_pkt_fwd",

```

```

    "min_seg_size_forward", "Active Mean", "Active Std", "Active Max", "Active
Min",
    "Idle Mean", "Idle Std", "Idle Max", "Idle Min", "Label"]

ths = open("importance_list_for_attack_files.csv", "w")
folder("./feature_pics/")
for j in csv_files:
    df=pd.read_csv(".\\attacks\\"+j,usecols=main_labels)
    df=df.fillna(0)
    attack_or_not=[]
    for i in df["Label"]:#it changes the normal label to "1" and the attack
tag to "0" for use in the machine learning algorithm
        if i == "BENIGN":
            attack_or_not.append(1)
        else:
            attack_or_not.append(0)
    df["Label"]=attack_or_not

    y = df["Label"].values
    del df["Label"]
    X = df.values

    X = np.float32(X)
    X[np.isnan(X)] = 0
    X[np.isinf(X)] = 0

    #computing the feature importances
    forest =
sk.ensemble.RandomForestRegressor(n_estimators=250,random_state=0)
    forest.fit(X, y)
    importances = forest.feature_importances_
    std = np.std([tree.feature_importances_ for tree in forest.estimators_],
axis=0)
    indices = np.argsort(importances)[::-1]
    refclasscol=list(df.columns.values)
    impor_bars =
pd.DataFrame({'Features':refclasscol[0:20],'importance':importances[0:20]})
    impor_bars =
impor_bars.sort_values('importance',ascending=False).set_index('Features')
    plt.rcParams['figure.figsize'] = (10, 5)
    impor_bars.plot.bar();
    #printing the feature importances
    count=0
    fea_ture=j[0:-4]+"=["
    for i in impor_bars.index:
        fea_ture=fea_ture+"\""+str(i)+"\", "
        count+=1
        if count==5:
            fea_ture=fea_ture[0:-1]+"]"
            break
    print(j[0:-4], "importance list:")
    print(j[0:-4], "\n", impor_bars.head(20), "\n\n\n")
    print(fea_ture)
    plt.title(j[0:-4]+" Attack - Feature Importance")
    plt.ylabel('Importance')

```

```

plt.savefig("./feature_pics/"+j[0:-4]+".pdf",bbox_inches='tight',
papertype = 'a4', orientation = 'portrait', format = 'pdf')
ths.write(( fea_ture ) )
plt.tight_layout()
plt.show()
print("-----\n\n\n\n")

print("mission accomplished!")
print("Total operation time: = ",time.time()- seconds ,"seconds")
ths.close()

```

feature_selection_for_all_data.ipynb

```

## "all_data.csv" file is required for the operation of the program.
## "all_data.csv" file must be located in the same directory as the program.

## the purpose of this code is to determine which features to use in the
machine learning phase.
## for this purpose, the importance weights of the attacks are calculated.
## this calculation was made using sklearn-RandomForestRegressor.

## the some codes parts used for calculation and graphing are taken from the
following site.
## http://scikit-
learn.org/stable/auto_examples/ensemble/plot_forest_importances.html

import numpy as np
import os
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import RandomForestRegressor
import sklearn as sk
import time
seconds = time.time()

def folder(f_name): #this function creates a folder named "feature_pics" in
the program directory.
    try:
        if not os.path.exists(f_name):
            os.makedirs(f_name)
    except OSError:
        print ("The folder could not be created!")

# CSV files names:
csv_files=["all_data.csv"]# It creates a list of file names in the "attacks"
folder.

```

```

# Headers of column
main_labels=["Flow Duration","Total Fwd Packets",    "Total Backward
Packets","Total Length of Fwd Packets","Total Length of Bwd Packets","Fwd
Packet Length Max","Fwd Packet Length Min",
    "Fwd Packet Length Mean","Fwd Packet Length Std","Bwd Packet Length
Max","Bwd Packet Length Min","Bwd Packet Length Mean","Bwd Packet Length
Std",
    "Flow Bytes/s","Flow Packets/s","Flow IAT Mean","Flow IAT Std","Flow IAT
Max","Flow IAT Min","Fwd IAT Total","Fwd IAT Mean","Fwd IAT Std","Fwd IAT
Max",
    "Fwd IAT Min","Bwd IAT Total","Bwd IAT Mean","Bwd IAT Std","Bwd IAT
Max","Bwd IAT Min","Fwd PSH Flags","Bwd PSH Flags","Fwd URG Flags","Bwd URG
Flags",
    "Fwd Header Length","Bwd Header Length","Fwd Packets/s","Bwd
Packets/s","Min Packet Length","Max Packet Length","Packet Length
Mean","Packet Length Std",
    "Packet Length Variance","FIN Flag Count","SYN Flag Count","RST Flag
Count","PSH Flag Count","ACK Flag Count","URG Flag Count","CWE Flag Count",
    "ECE Flag Count","Down/Up Ratio","Average Packet Size","Avg Fwd Segment
Size","Avg Bwd Segment Size","Fwd Avg Bytes/Bulk",
    "Fwd Avg Packets/Bulk","Fwd Avg Bulk Rate","Bwd Avg Bytes/Bulk","Bwd Avg
Packets/Bulk","Bwd Avg Bulk Rate","Subflow Fwd Packets","Subflow Fwd Bytes",
    "Subflow Bwd Packets","Subflow Bwd
Bytes","Init_Win_bytes_forward","Init_Win_bytes_backward","act_data_pkt_fwd",
    "min_seg_size_forward","Active Mean","Active Std","Active Max","Active
Min",
    "Idle Mean","Idle Std","Idle Max", "Idle Min","Label"]

ths = open("importance_list_all_data.csv", "w")
folder("./feature_pics/")
for j in csv_files:
    df=pd.read_csv(j,usecols=main_labels)
    df=df.fillna(0)
    attack_or_not=[]
    for i in df["Label"]:#it changes the normal label to "1" and the attack
tag to "0" for use in the machine learning algorithm
        if i == "BENIGN":
            attack_or_not.append(1)
        else:
            attack_or_not.append(0)
    df["Label"]=attack_or_not

    y = df["Label"].values
    del df["Label"]
    X = df.values

    X = np.float32(X)
    X[np.isnan(X)] = 0
    X[np.isinf(X)] = 0

    #computing the feature importances
    forest =
sk.ensemble.RandomForestRegressor(n_estimators=250,random_state=0)
    forest.fit(X, y)

```

```

importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_],
              axis=0)
indices = np.argsort(importances)[::-1]
refclasscol=list(df.columns.values)
impor_bars =
pd.DataFrame({'Features':refclasscol[0:20],'importance':importances[0:20]})
impor_bars =
impor_bars.sort_values('importance',ascending=False).set_index('Features')
plt.rcParams['figure.figsize'] = (10, 5)
impor_bars.plot.bar();
#printing the feature importances
count=0
fea_ture=j[0:-4]+"=["
for i in impor_bars.index:
    fea_ture=fea_ture+"\""+str(i)+"\", "
    count+=1
    if count==5:
        fea_ture=fea_ture[0:-1]+"]"
        break
print(j[0:-4],"importance list:")
print(j[0:-4],"\\n",impor_bars.head(20),"\\n\\n\\n")
print(fea_ture)
plt.title(j[0:-4]+" Attack - Feature Importance")
plt.ylabel('Importance')
plt.savefig("./feature_pics/"+j[0:-4]+".pdf",bbox_inches='tight',
papertype = 'a4', orientation = 'portrait', format = 'pdf')
ths.write(( fea_ture ) )
plt.tight_layout()
#plt.show()
print("-----\\n\\n\\n\\n")

print("mission accomplished!")
print("Total operation time: = ",time.time()- seconds ,"secomds")
ths.close()

```

