Intorsion detection with AI

By Sarah Jabr

Supervised by Ruba Alkusheiny

Table of Contents

The Dataset

This database contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment.

Feature Table

Nr	Feature	Description
1	duration	Duration of connection in seconds
2	protocol_type	Connection protocol (tcp, udp, icmp)
3	service_dst	Port mapped to service (e.g. http, ftp, ..)
4	flag	Normal or error status flag of connection
5	src_bytes	Number of data bytes from src to dst
6	dst_bytes	Number of data bytes transferred from destination to source
7	land	If source and destination port no. and IP addresses are same then it will set as 1 otherwise 0
8	wrong_fragment	Total number of wrong fragments in a connection
9	urgent	Number of urgent packets (these packets with urgent bit activated)
10	hot	Number of 'hot' indicators means entering in a system directory
11	num_failed_logins	Number of failed login attempts
12	logged_in	Shows login status (1- successful login, 0- otherwise)
13	num_compromised	Number of compromised conditions
14	root_shell	Shows root shell status (1-if root shell obtained otherwise 0)
15	su_attempted	Set as 1 if 'su root' command used otherwise set as 0
16	num_root	Number of operations performed as root
17	num_file_creations	Number of file creation operations
18	num_shells	Number of shell prompts in a connection
19	num_access_files	Number of operations on access control files
20	num_outbound_cmds	Number of outbound commands in a ftp session
21	is_host_login	If login as root or admin then this set as 1 otherwise 0
22	is_guest_login	Set as 1 if login as guest otherwise 0
23	count	Number of connections to the same destination host
24	srv_count	Number of connection to the same service (port number)
25	serror_rate	Percentage of connections that have activated flag (#4) s0,s1,s2 or s3, among the connections aggregated in count (#23)

26	srv_serror_rate	Percentage of connection that have activated flag (#4) s0,s1,s2 or s3, among the connections aggregated in srv count (#24)
27	rerror_rate	Percentage of connections that have activated flag (#4) REJ, among the connections aggregated in count (#23)
28	srv_rerror_rate	Percentage of connections that have activated flag (#4) REJ, among the connections aggregated in srv count (#24)
29	same_srv_rate	Percentage of connections that were to the same services, among the connections aggregated in count (#23)
30	diff_srv_rate	Percentage of connections that were to the different services, among the connections aggregated in count (#23)
31	srv_diff_host_rate	Percentage of connections that were to different destination machines among the connections aggregated in srv count (#24)
32	dst_host_count	Number of connections having the same destination host IP address
33	dst_host_srv_count	Number of connections having same port number
34	dst_host_same_srv_rate	Percentage of connections that were to the same service among the connections aggregated in dst host count (#32)
35	dst_host_diff_srv_rate	Percentage of connections that were to different service among the connections aggregated in dst host count (#32)
36	dst_host_same_src_port_rate	Percentage of connections that were to the same source port among the connections aggregated in dst host srv count (#33)
37	dst_host_srv_diff_host_rate	Percentage of connections that were to the different destination machines among the connections aggregated in dst host srv count (#33)
38	dst_host_serror_rate	Percentage of connections that have activated flag (#4) s0,s1,s2 or s3, among the connections aggregated in dst host count (#32)
39	dst_host_srv_serror_rate	Percentage of connections that have activated flag (#4) s0,s1,s2 or s3, among the connections aggregated in dst host srv count (#33)
40	dst_host_rerror_rate	Percentage of connections that have activated flag (#4) REJ, among the connections aggregated in dst host count (#32)
41	dst_host_srv_rerror_rate	Percentage of connections that have activated flag (#4) REJ, among the connections aggregated in dst host srv count (#32)
42	label	Attack class label

Data reading

Using **pd.read_csv() chnksize** parameter to read the data in chunks and **on_bad_lines** parameter to skip bad lines with a warning.

Then using **pd.concat()** to concatenate each chunk to the DataFrame **dataset.**

```
data=pd.read_csv('data.csv',on_bad_lines="warn",names=[i for i in range(42)],
                 engine='python',chunksize=10000)
dataset=pd.concat(data)
```

Data preprocessing

The data is very much clean. The following lines was used to check for mixed data type and null values.

```
print(dataset.isnull().sum(0))
print(dataset.info())
```

There were no null values however columns Nr 1,2,3 and 42 has an object data type wich means they are probably strings.

To check if these columns contains categorical data the following lines was used

```
print(dataset.nunique())
```

And the following results was returned:

```
1          3
2         70
3         11
```

```
41         23
```

These results confirms that these columns contain categorical data since we have huge amount of data but very few unique values in comparison.

In column 41 which contains the label we have the dependent value which determines if a connection is normal or is an attack

```
dataset[41].unique()
```

```
['normal.', 'buffer_overflow.', 'loadmodule.', 'perl.', 'neptune.',
 'smurf.', 'guess_passwd.', 'pod.', 'teardrop.', 'portsweep.',
 'ipsweep.', 'land.', 'ftp_write.', 'back.', 'imap.', 'satan.',
 'phf.', 'nmap.', 'multihop.', 'warezmaster.', 'warezclient.',
 'spy.', 'rootkit.'], dtype=object)
```

Each attack from the above list falls into one of the four main categories:

- DOS: denial-of-service, e.g. syn flood;
- R2L: unauthorized access from a remote machine, e.g. guessing password;
- U2R: unauthorized access to local superuser (root) privileges, e.g., various ``buffer overflow'' attacks;
- probing: surveillance and other probing, e.g., port scanning.

Before dealing with the object data type the data was sectioned into dependent and independent values.

```
X=dataset.iloc[:,:-1].values
y=dataset.iloc[:,-1]
```

Then label encoding y to turn each category in y into a label.

```
from sklearn.preprocessing import LabelEncoder
labelencoder=LabelEncoder()
y=labelencoder.fit_transform(y)
```

Using get dummies to encode column 1,2 and 3.

```
X1=pd.get_dummies(X[:,1],drop_first=True)
X2=pd.get_dummies(X[:,2],drop_first=True)
X3=pd.get_dummies(X[:,3],drop_first=True)
```

Delete the original columns.

```
X=np.delete(X,[1,2,3], axis=1)
```

Attach the new encoded ones.

```
np.append(X,X1,axis=1)
np.append(X,X2,axis=1)
np.append(X,X3,axis=1)
```

Intrusion Detector Learning

Software to detect network intrusions protects a computer network from unauthorized users, including perhaps insiders. The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between bad'' connections, called intrusions or attacks, and good'' normal connections.

Split the data into training and testing sets

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=1/4,
                                                random_state=42)
```

Scale the independent values

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)#you use fit only once for each scaler
X_test = sc.transform(X_test)
```

Train a Logistic Regression Model

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(X_train, y_train)
```

The results

Test the Model

```
y_pred = classifier.predict(X_test)
```

Calculate the results' accuracy

```
import sklearn.metrics as ms
print('Acuracy=',ms.accuracy_score(y_test, y_pred))
print("Mean absolute error =", round(ms.mean_absolute_error(y_test,y_pred), 3))
print("Mean squared error =", round(ms.mean_squared_error(y_test,y_pred), 3))
print("Median absolute error =", round(ms.median_absolute_error(y_test,y_pred), 3))
print("Explain variance score =", round(ms.explained_variance_score(y_test,y_pred), 3))
print("R2 score =", round(ms.r2_score(y_test, y_pred), 3))
```

```
Acuracy= 0.9988265632757585
Mean absolute error = 0.007
Mean squared error = 0.055
Median absolute error = 0.0
Explain variance score = 0.997
R2 score = 0.997
```