Este conjunto de dados de Rede tem 2 Classes, uma é Normal e outra é Anomalia.

Estas são as coisas que você pode tentar neste conjunto de dados:

1) O principal objetivo é detectar a anomalia usando dados rotulados.

2) Também tente detectar os padrões nos dados normais e de anomalia sem usar dados rotulados, utilizando métodos não supervisionados.

3) Também tente criar um modelo que detecte comportamento anormal em cada sistema, se possível.

Métricas de Desempenho:

- Acurácia

- Precisão (Ponderada, micro, macro)

- Recall

- ROC

- AUC

- F1-Score (Ponderado, micro, macro)

- Recall

- Sensibilidade

- Relatório de Classificação

- Métrica personalizada

Características do Conjunto de Dados:

Destination Port	Flow Duration	Total FwdPackets	Total Backward Packets
Total Length of Fwd Packets	Total Length of BwdPackets	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 IATMean	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 PSHFlags	Bwd PSH Flags
Fwd URGFlags	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	FINFlag Count
SYNFlag Count	RSTFlag Count	PSH Flag Count	ACK Flag Count
URG Flag Count	CWEFlag Count	ECEFlag Count	Down/Up Ratio
Average Packet Size	Avg Fwd Segment Size	Avg Bwd Segment Size	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		

Atividades Normais e de Ataque de cada arquivo:

Name of Files	Day Activity	Attacks Found
Monday-WorkingHours.pcap_ISCX.csv	Monday	Benign (Normal human activities)
Tuesday-WorkingHours.pcap_ISCX.csv	Tuesday	Benign, FTP-Patator, SSH-Patator
Wednesday-workingHours.pcap_ISCX.csv	Wednesday	Benign, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS slowloris, Heartbleed
Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv	Thursday	Benign, Web Attack – Brute Force, Web Attack – Sql Injection, Web Attack – XSS
Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv	Thursday	Benign, Infiltration
Friday-WorkingHours-Morning.pcap_ISCX.csv	Friday	Benign, Bot
Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv	Friday	Benign, PortScan
Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv	Friday	Benign, DDoS

Segunda-feira, 3 de julho de 2017

Benigno (Atividades humanas normais)

Terça-feira, 4 de julho de 2017

Força Bruta

FTP-Patator (9:20 – 10:20 a.m.)

SSH-Patator (14:00 – 15:00 p.m.)

Atacante: Kali, 205.174.165.73

Vítima: WebServer Ubuntu, 205.174.165.68 (IP Local: 192.168.10.50)

Processo NAT no Firewall:

Ataque: 205.174.165.73 -> 205.174.165.80 (IP válido do Firewall) -> 172.16.0.1 -> 192.168.10.50

Resposta: 192.168.10.50 -> 172.16.0.1 -> 205.174.165.80 -> 205.174.165.73

Quarta-feira, 5 de julho de 2017

DoS / DDoS

DoS slowloris (9:47 – 10:10 a.m.)

DoS Slowhttptest (10:14 – 10:35 a.m.)

DoS Hulk (10:43 – 11 a.m.)

DoS GoldenEye (11:10 – 11:23 a.m.)

Atacante: Kali, 205.174.165.73

Vítima: WebServer Ubuntu, 205.174.165.68 (IP Local: 192.168.10.50)

Processo NAT no Firewall:

Ataque: 205.174.165.73 -> 205.174.165.80 (IP válido do Firewall) -> 172.16.0.1 -> 192.168.10.50

Resposta: 192.168.10.50 -> 172.16.0.1 -> 205.174.165.80 -> 205.174.165.73

Heartbleed Port 444 (15:12 - 15:32)

Atacante: Kali, 205.174.165.73

Vítima: Ubuntu12, 205.174.165.66 (IP Local: 192.168.10.51)

Processo NAT no Firewall:

Ataque: 205.174.165.73 -> 205.174.165.80 (IP válido do Firewall) -> 172.16.0.11 -> 192.168.10.51

Resposta: 192.168.10.51 -> 172.16.0.1 -> 205.174.165.80 -> 205.174.165.73

Quinta-feira, 6 de julho de 2017

Manhã

Ataque Web – Força Bruta (9:20 – 10 a.m.)

Ataque Web – XSS (10:15 – 10:35 a.m.)

Ataque Web – Injeção SQL (10:40 – 10:42 a.m.)

Atacante: Kali, 205.174.165.73

Vítima: WebServer Ubuntu, 205.174.165.68 (IP Local: 192.168.10.50)

Processo NAT no Firewall:

Ataque: 205.174.165.73 -> 205.174.165.80 (IP válido do Firewall) -> 172.16.0.1 -> 192.168.10.50

Resposta: 192.168.10.50 -> 172.16.0.1 -> 205.174.165.80 -> 205.174.165.73

Tarde

Infiltração – Download no Dropbox

Meta exploit Win Vista (14:19 e 14:20-14:21 p.m.) e (14:33 -14:35)

Atacante: Kali, 205.174.165.73

Vítima: Windows Vista, 192.168.10.8

Infiltração – Disco Cool – MAC (14:53 p.m. – 15:00 p.m.)

Atacante: Kali, 205.174.165.73

Vítima: MAC, 192.168.10.25

Infiltração – Download no Dropbox

Win Vista (15:04 – 15:45 p.m.)

Primeira Etapa:

Atacante: Kali, 205.174.165.73

Vítima: Windows Vista, 192.168.10.8

Segunda Etapa (Portscan + Nmap):

Atacante: Vista, 192.168.10.8

Vítima: Todos os outros clientes

Sexta-feira, 7 de julho de 2017

Manhã

Botnet ARES (10:02 a.m. – 11:02 a.m.)

Atacante: Kali, 205.174.165.73

Vítimas: Win 10, 192.168.10.15 + Win 7, 192.168.10.9 + Win 10, 192.168.10.14 + Win 8, 192.168.10.5 + Vista, 192.168.10.8

Tarde

Port Scan:

Regras do Firewall ligadas (13:55 – 13:57, 13:58 – 14:00, 14:01 – 14:04, 14:05 – 14:07, 14:08 - 14:10, 14:11 – 14:13, 14:14 – 14:16, 14:17 – 14:19, 14:20 – 14:21, 14:22 – 14:24, 14:33 – 14:33, 14:35 - 14:35)

Regras do Firewall desligadas (sS 14:51-14:53, sT 14:54-14:56, sF 14:57-14:59, sX 15:00-15:02, sN 15:03-15:05, sP 15:06-15:07, sV 15:08-15:10, sU 15:11-15:12, sO 15:13-15:15, sA 15:16-15:18, sW 15:19-15:21, sR 15:22-15:24, sL 15:25-15:25, sI 15:26-15:27, b 15:28-15:29)

Atacante: Kali, 205.174.165.73

Vítima: Ubuntu16, 205.174.165.68 (IP Local: 192.168.10.50)

Processo NAT no Firewall:

Atacante: 205.174.165.73 -> 205.174.165.80 (IP válido do Firewall) -> 172.16.0.1

Tarde

DDoS LOIT (15:56 – 16:16)

Atacantes: Três Win 8.1, 205.174.165.69 - 71

Vítima: Ubuntu16, 205.174.165.68 (IP Local: 192.168.10.50)

Processo NAT no Firewall:

Atacantes: 205.174.165.69, 70, 71 -> 205.174.165.80 (IP válido do Firewall) -> 172.16.0.1

Total de instancias de ataques e atividades normais

Class Labels	Number of instances
BENIGN	2359087
DoS Hulk	231072
PortScan	158930
DDoS	41835
DoS GoldenEye	10293
FTP-Patator	7938
SSH-Patator	5897
DoS slowloris	5796
DoS Slowhttptest	5499
Botnet	1966
Web Attack – Brute Force	1507
Web Attack – XSS	652
Infiltration	36
Web Attack – Sql Injection	21
Heartbleed	11

CODIGO

Este código realiza a detecção de intrusões em uma rede usando um conjunto de dados e redes neurais artificiais (ANN). Aqui está uma explicação detalhada das etapas envolvidas no código:

1. **Carregamento e visualização dos dados**

```python
df = pd.read_csv('/kaggle/input/network-intrusion-dataset/Friday-WorkingHours-Morning.pcap_ISCX.csv')
df.head()
df.columns
```

O código carrega um arquivo CSV contendo dados de tráfego de rede e exibe as primeiras linhas do DataFrame para entender a estrutura dos dados.

2. **Codificação de rótulos e pré-processamento**

```python
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
df[' Label'] = encoder.fit_transform(df[' Label'])  # Transformação dos rótulos em valores numéricos
df = df.fillna(0)  # Substitui valores NaN por 0
df = df.replace([np.inf, -np.inf], 0)  # Substitui valores infinitos por 0
df.isnull().sum()
df = df.astype(int)  # Converte os valores do DataFrame em inteiros
```

- O rótulo das intrusões (`' Label'`) é transformado de texto para valores numéricos usando o `LabelEncoder`.

- Valores nulos (NaN) e infinitos são substituídos por zero, e o DataFrame é convertido para tipos inteiros para simplificação.

3. **Seleção de características (Feature Selection)**

```python
from sklearn.feature_selection import SelectKBest, f_classif
```

```
X = df.drop(' Label', axis=1)

y = df[' Label']

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)

imputer = SimpleImputer(strategy='mean')

X_imputed = imputer.fit_transform(X)

k_best = SelectKBest(score_func=f_classif, k=10)

X_new = k_best.fit_transform(X_imputed, y)

selected_feature_names = X.columns[k_best.get_support()]
```

- Aqui, usamos `SelectKBest` para selecionar as 10 melhores características com base na análise estatística `f_classif`.

- O DataFrame é escalonado e valores ausentes são imputados (substituídos pela média).

- As 10 melhores características são extraídas e armazenadas.

4. **Criação de um novo DataFrame com as melhores características**

```python
new_columns = [' Destination Port', ' Bwd Packet Length Min', ... ]

df_new = X[new_columns]

df_new['label'] = df[' Label']
```

Um novo DataFrame `df_new` é criado com as 10 melhores características, além da coluna de rótulos.

5. **Divisão dos dados em treinamento e teste**

```python
X1 = df_new.iloc[:,:-1].values

y1 = df_new.iloc[:,-1].values

X_train, X_test, y_train, y_test = train_test_split(X1, y1, test_size=0.3, random_state=42)
```

Os dados são divididos em conjuntos de treinamento e teste, com 70% para treinamento e 30% para teste.

6. **Criação e treinamento de uma Rede Neural Artificial (ANN)**

```python
import tensorflow as tf
ann = tf.keras.models.Sequential()
ann.add(tf.keras.layers.Dense(units=10, activation='sigmoid'))
ann.add(tf.keras.layers.Dense(units=10, activation='sigmoid'))
ann.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
ann.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
ann.fit(X_train, y_train, batch_size=32, epochs=10)
```

- Uma ANN sequencial é construída com 3 camadas densas (fully connected).
- As ativações de todas as camadas são `sigmoid`, e o otimizador escolhido é `adam`.
- O treinamento é feito com os dados de treinamento, utilizando `EarlyStopping` para interromper o treinamento quando o modelo não melhorar após 10 épocas.

7. **Fazendo previsões com a rede neural**

```python
print(ann.predict([[3268,72,72,0,0,0,0,201,72,32]]))
```

O modelo faz previsões com base em um conjunto de dados fictício fornecido (10 características).

8. **Treinamento de um modelo de regressão linear para comparação**

```python
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
from sklearn.metrics import mean_squared_error, r2_score
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R-squared:", r2_score(y_test, y_pred))
```

```

- Um modelo de regressão linear também é treinado para prever os rótulos com base nas características selecionadas.

- Métricas como o erro quadrático médio (MSE) e o coeficiente de determinação ($R^2$) são usadas para avaliar o desempenho do modelo.
```