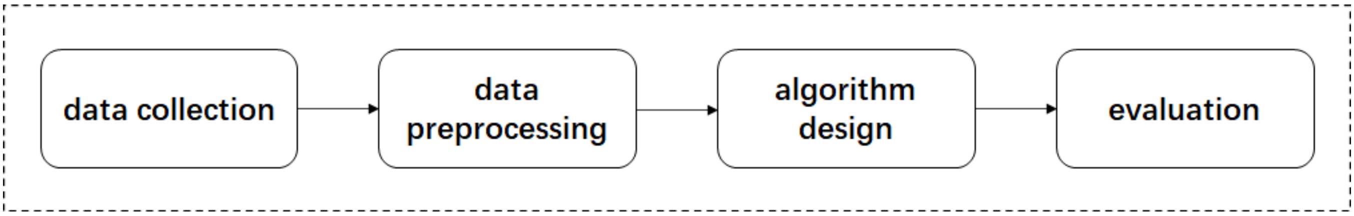
机器学习流程如下：



data collection

采用公开数据集：https://www.unb.ca/cic/datasets/

data preprocessing

使用 Joy 完成数据预处理工作，通过处理原始 .pcap 数据包，得到 JSON 压缩文件，其中存储了关于流量的元数据。

解压缩得到 JSON 文件，从 JSON 文件中可以提取出 3 类元数据特征，分别为：

- 字节分布
- 包长度序列转移概率矩阵 / 包长度序列
- 时间间隔序列转移概率矩阵 / 时间间隔序列

1、提取字节分布特征

```
def getByteDistribution(self):

    if self.flows == []:
        return None

    data = []

    for flow in self.flows:
        if len(flow['packets']) == 0:
            continue
        if 'byte_dist' in flow and sum(flow['byte_dist']) > 0:
            tmp = map(lambda x: x/float(sum(flow['byte_dist'])),flow['byte_dist'])
            data.append(tmp)
        else:
            data.append(np.zeros(256))

    return data
```

仅统计 TCP/UDP 负载前 30 字节的字节分布概率。

2、提取包长度序列转移概率矩阵 / 包长度序列

```
def getIndividualFlowPacketLengths(self):

    if self.flows == []:
        return None

    data = []

    if self.compact:
        numRows = 10
        binSize = 150.0
    else:
        numRows = 60
        binSize = 25.0

    for flow in self.flows:
        transMat = np.zeros((numRows,numRows))
        if len(flow['packets']) == 0:
            continue
```

```
        elif len(flow['packets']) == 1:
            curPacketSize = min(int(flow['packets'][0]['b']/binSize),numRows-1)
            transMat[curPacketSize,curPacketSize] = 1
            data.append(list(transMat.flatten()))
            continue

        # get raw transition counts
        for i in range(1,len(flow['packets'])):
            prevPacketSize = min(int(flow['packets'][i-1]['b']/binSize),numRows-1)
            if 'b' not in flow['packets'][i]:
                break
            curPacketSize = min(int(flow['packets'][i]['b']/binSize),numRows-1)
            transMat[prevPacketSize,curPacketSize] += 1

        # get empirical transition probabilities
        for i in range(numRows):
            if float(np.sum(transMat[i:i+1])) != 0:
                transMat[i:i+1] = transMat[i:i+1]/float(np.sum(transMat[i:i+1]))

        data.append(list(transMat.flatten()))

    return data
```

仅统计最多前 200 个数据包的长度转移概率矩阵。

```
def getIndividualFlowPacketLengths_seq(self):

    if self.flows == []:
        return None

    data = []
    num = 10
    binSize = 8

    for flow in self.flows:
        lenSeq = np.zeros(num)
        if len(flow['packets']) == 0:
            continue

        for i in range(0, min(num, len(flow['packets']))):
            if 'b' not in flow['packets'][i]:
                lenSeq[i] = 0
                continue
            lenSeq[i] = int(flow['packets'][i]['b']/binSize)

        data.append(list(lenSeq.flatten()))

    return data
```

仅选取每个流的前 10 个数据包长度组成序列，并且设置 binSize 为 8。

3、提取时间间隔序列转移概率矩阵 / 时间间隔序列

```
def getIndividualFlowIPTs(self):

    if self.flows == []:
        return None

    data = []

    if self.compact:
        numRows = 10
        binSize = 50.0
    else:
        numRows = 30
        binSize = 50.0

    for flow in self.flows:
        transMat = np.zeros((numRows,numRows))
        if len(flow['packets']) == 0:
```

```
            continue
        elif len(flow['packets']) == 1:
            curIPT = min(int(flow['packets'][0]['ipt']/float(binSize)),numRows-1)
            transMat[curIPT,curIPT] = 1
            data.append(list(transMat.flatten()))
            continue

        # get raw transition counts
        for i in range(1,len(flow['packets'])):
            prevIPT = min(int(flow['packets'][i-1]['ipt']/float(binSize)),numRows-1)
            curIPT = min(int(flow['packets'][i]['ipt']/float(binSize)),numRows-1)
            transMat[prevIPT,curIPT] += 1

        # get empirical transition probabilities
        for i in range(numRows):
            if float(np.sum(transMat[i:i+1])) != 0:
                transMat[i:i+1] = transMat[i:i+1]/float(np.sum(transMat[i:i+1]))

        data.append(list(transMat.flatten()))

    return data
```

仅统计最多前 200 个数据包的时间间隔转移概率矩阵。

```
def getIndividualFlowIPTs_seq(self):

    if self.flows == []:
        return None

    data = []
    num = 10
    binSize = 8

    for flow in self.flows:
        lenSeq = np.zeros(num)
        if len(flow['packets']) == 0:
            continue

        for i in range(0, min(num, len(flow['packets']))):
            if 'ipt' not in flow['packets'][i]:
                lenSeq[i] = 0
                continue
            lenSeq[i] = int(flow['packets'][i]['ipt']/binSize)

        data.append(list(lenSeq.flatten()))

    return data
```

仅选取每个流的前 10 个时间间隔组成序列，并且设置 binSize 为 8。

algorithm design

朴素贝叶斯：

```
from sklearn.naive_bayes import GaussianNB

model_1 = GaussianNB()
```

k 近邻算法：

```
from sklearn.neighbors import KNeighborsClassifier

model_2 = KNeighborsClassifier()
```

决策树：

```
from sklearn import tree

model_3 = tree.DecisionTreeClassifier()
```

随机森林：

```
from sklearn.ensemble import RandomForestClassifier

model_4 = RandomForestClassifier(n_estimators=10, random_state=0)
```

支持向量机：

```
from sklearn.svm import LinearSVC

model_5 = LinearSVC(C=0.1, max_iter=1000, random_state=0)
```

evaluation

采用“10 折交叉验证”计算模型的各类评估指标。

```
y_pred = cross_val_predict(model, X, y, cv=10)

print(accuracy_score(y, y_pred))
print(confusion_matrix(y, y_pred))
print(precision_score(y, y_pred))
print(recall_score(y, y_pred))
```

实验一：使用“长度”相关特征对普通流量和 Tor 流量进行二分类

原始数据集中，正样本数量为 13006，负样本数量为 7644。

```
rows_1 = [2000, 1225, 967, 2000, 2814, 2000, 2000]
rows_2 = [1797, 208, 273, 1725, 598, 1559, 1484]
```

accuracy：

	method_1	**method_2**	**method_3**	**method_4**	**method_5**
len	52.9%	95.4%	96.3%	96.6%	94.9%
len_seq	84.0%	96.2%	98.3%	98.6%	76.9%

confusion matrix：

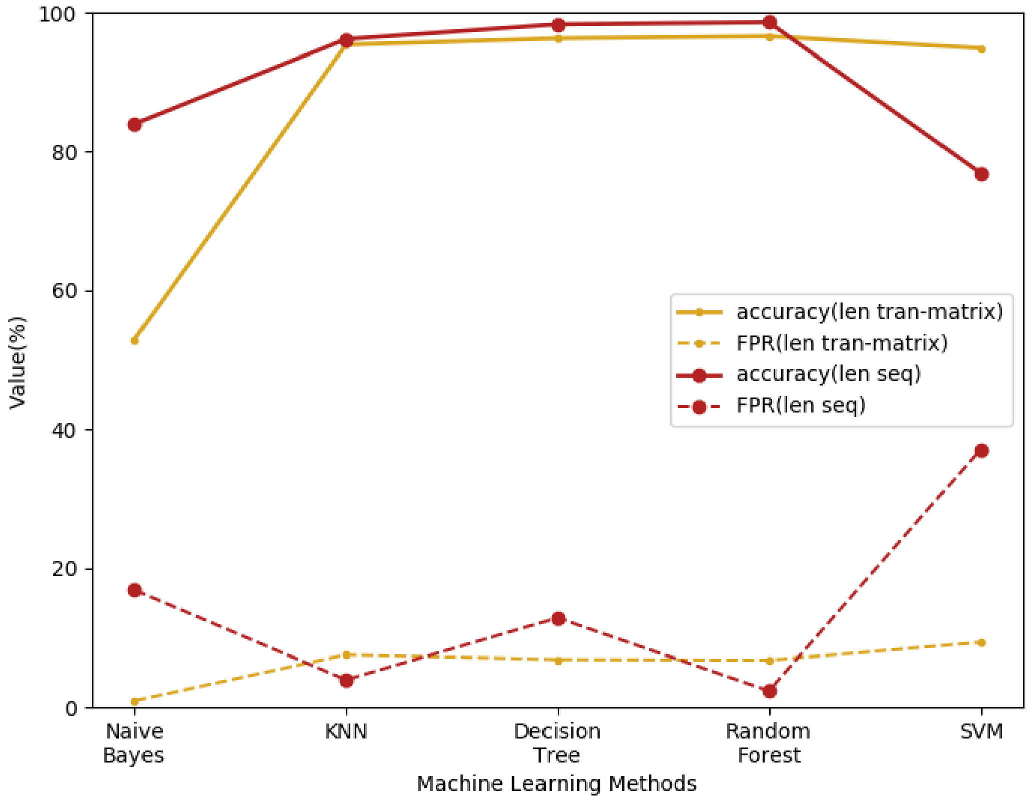
	method_1	**method_2**	**method_3**	**method_4**	**method_5**
len	7579 65 9659 3347	7068 576 374 12632	7122 522 247 12759	7158 486 214 12792	6932 712 336 12670
len_seq	6357 1287 2007 10999	7349 295 492 12514	7434 210 138 12868	7467 177 120 12886	4810 2834 1945 11061

precision：

	method_1	**method_2**	**method_3**	**method_4**	**method_5**
len	98.1%	95.6%	96.1%	96.3%	94.7%
len_seq	89.5%	97.7%	98.4%	98.6%	79.6%

recall：

	method_1	method_2	method_3	method_4	method_5
len	25.7%	97.1%	98.1%	98.4%	97.4%
len_seq	84.6%	96.2%	98.9%	99.1%	85.0%



实验二：使用“时间”相关特征对普通流量和 Tor 流量进行二分类

正样本数量为 13006，负样本数量为 7644。

```
rows_1 = [2000, 1225, 967, 2000, 2814, 2000, 2000]
rows_2 = [1797, 208, 273, 1725, 598, 1559, 1484]
```

accuracy：

	method_1	method_2	method_3	method_4	method_5
ipt	74.7%	77.1%	82.7%	84.1%	81.9%
ipt_seq	63.7%	78.2%	81.9%	84.1%	62.0%

confusion matrix：

	method_1	method_2	method_3	method_4	method_5
len	3681 3963 1270 11736	5360 2284 2449 10557	5180 2464 1095 11911	5407 2237 1052 11954	5131 2513 1230 11776
len_seq	591 7053 433 12573	5033 2611 1887 11119	5101 2543 1203 11803	5654 1990 1302 11704	1842 5802 2051 10955

precision：

	method_1	method_2	method_3	method_4	method_5
ipt	74.8%	82.2%	82.9%	84.2%	82.4%

ipt_seq	64.1%	81.0%	82.3%	85.5%	65.4%

recall:

	method_1	**method_2**	**method_3**	**method_4**	**method_5**
ipt	90.2%	81.2%	91.6%	91.9%	90.5%
ipt_seq	96.7%	85.5%	90.8%	90.0%	84.2%