---------Checar bibliotecas---------
baixarBibliotecas.sh
Baixa as bibliotecas necessárias.

checaBibliotecas.py
Checa se tem as bibliotecas necessárias para a execução correta. É necessário todas elas para o correto funcionamento.

---------Tratamento de log---------
run-analysisALL.sh
Script que seleciona todos os pcap de diferentes dias e salvam apenas o tipo específico que quero ('amazonEcho', 'babyMonitor', 'belkinMotion', 'blipcareBP', 'laptop', 'lifxlighbulb', 'netatmo', 'sleepSensor', 'tribySpeaker').

Junto os pcap de dias diferentes em 1 só usando comando básico em shell/Linux.

Separo em .csvs diferentes

AdicionarLabelFlow.py
Pego cada flow total diferente e adiciono uma nova coluna especificando o tipo de tráfego ('amazonEcho', 'babyMonitor', 'belkinMotion', 'blipcareBP', 'laptop', 'lifxlighbulb', 'netatmo', 'sleepSensor', 'tribySpeaker') e salvo em um arquivo (logFlow.csv por exemplo).

mediaFlow.py
Adiciono colunas que calculem a média, mínimo, máximo e variância e salvo em um arquivo de saída (logFlowLabel.csv)

----------Execução----------

handoutFluxo.py
Importa as bibliotecas, atribui nome para as colunas usando biblioteca pandas. Coloco o tamanho e atribuo o seed, divido o dataset em treino, teste e validação, além de especificar os nomes das classes do dataset ('amazonEcho', 'babyMonitor', 'belkinMotion', 'blipcareBP', 'laptop', 'lifxlighbulb', 'netatmo', 'sleepSensor', 'tribySpeaker'). Faço um laço *for* instanciando os modelos de aprendizado de máquina (NB, CART, R. Forest, Bagging, k-NN, SVM, K-means, Adaboost) e retiro as diferentes métricas (accuracy, f1 score, recall, precision, tempo de classificação) e guardo em vetores temporários. No final, gravo todos os resultados guardados nos vetores em um arquivo de saída (por exemplo, o resultadosFlowHoldout.txt ou ResultadosHoldoutPacotes.txt) usando paralelização de processos.

kfoldFluxo.py
É o mesmo funcionamento/ideia do handoutFluxo.py, exceto que neste nós vamos trabalhar com o K-Fold ao invés de trabalhar com treino, teste, validação. Portanto, é o próprio sistema que gerará a divisão do dataset. No final, salvamos em um arquivo de saída (resultadosFlowKFold.txt ou resultadosKfoldPacotes.txt).

Arquivo Python – salva em resultadosFlowHoldout.txt

No geral, teremos 4 programas em Python: 2 para Flow Scenario e 2 para Packet Scenario. Especificamente, FlowHoldoutScenario, FlowKFoldScenario, PacketHoldoutScenario, PacketKFoldScenario. Cada 1 destes gerará 1 log de resultado, respectivamente:

resultadosFlowHoldout.txt, resultadosFlowKFold.txt, resultadosHoldoutPacotes.txt, resultadosKFoldPacotes.txt.

Portanto, no total, teremos 4 programas python, 4 cenários, 4 resultados. A seguir, continuarei apenas para 1 resultado. Mas teremos que fazer o mesmo para os 3 outros cenários.

resultadosFlowHoldout.txt
Aqui está contido todo o log de resultado/saída.

Tratamento no awk,sed
Pego as colunas certas, por tipo, separadas em vírgulas e salvo apenas o essencial no arquivo (zum3.txt). Eu seleciono todas as primeiras colunas (accuracy), ou segundas (f1), ou terceiras (recall), ou quartas (precision), ou quintas (tempo). Nem tudo está mostrado aqui, mas pode fazer de outros modos.

```
grep 'CART' resultadosKfoldPacotes.txt | awk '{print $5}'| sed 's/,$//g' > f1CART.txt
grep 'NB' resultadosKfoldPacotes.txt | awk '{print $5}'| sed 's/,$//g' > f1NB.txt
paste -d ' ' f1NB.txt f1CART.txt f1RForest.txt f1Bagging.txt f1KNN.txt f1SVM.txt f1AdaBoost.txt > zum.txt
awk '{print $1*100,$2*100,$3*100,$4*100,$5*100,$6*100,$7*100}' zum.txt > zum2.txt
sed -e "s/ /,/g" < zum2.txt > zum3.txt
```

std.py
- Lê arquivo de texto (zum3.txt), pega as 7 colunas originais de valores (NB, CART, R. Forest, Bagging, k-NN, SVM, Adaboost) e salva em um .csv colocando as colunas originais separadas por 2 colunas com seus desvios padrões (standard deviation). Portanto, terei 5 .csv, ou seja 1 de cada métrica diferente (1 de accuracy, 1 de f1_score, 1 de precision, 1 de recall e 1 de tempo). Assim terei os 5 .csv para o cenário Flow Holdout: DatasetAccuracyHoldoutFlow.csv, DatasetF1HoldoutFlow.csv, DatasetPrecisionHoldoutFlow.csv, DatasetRecallHoldoutFlow.csv, DatasetTimeHoldoutFlow.csv,

oplot.gp
Programa em Gnuplot para plotar os resultados. Plotarei 5 gráficos (1 para cada métrica).

----------Comentários----------
Analisando: 1 programa, 1 cenário, 1 log de resultado. 5 csv. 5 plots.
No final, então teremos 4 programas, 4 cenários, 4 log de resultados, 20 csv, 20 plots.

----------Informações Extras----------

Tirar confusion matrix
```
cm = confusion_matrix(Y_validation, predictions)
cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
print(cm.diagonal())
```

detalhesSobreLogsPacotes.txt
Log opcional para tomar maior conhecimento sobre o dataset avaliado no cenário de pacotes. Para executá-lo, use:
```
import pandas
df = pandas.read_csv("./logFlowLabel.csv", sep=",", header='infer')
print(df.groupby('type')[['packets','average','min','max','vari']].describe())
```

DescricaoFluxo.txt

Log opcional para tomar maior conhecimento sobre o dataset avaliado no cenário de fluxos. Para executá-lo, use:

```
import pandas
df = pandas.read_csv("./logFlowLabel.csv", sep=",", header='infer')
print(df.groupby('type')[['bytes_out','packets','average','min','max','vari']].describe())
```