```
import numpy as pd
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
```

1. Charger les donnnees

```
traffic_reseau = pd.read_csv('dataset_sdn.csv')

traffic_reseau
```

2. Analyse de la dataset

```
traffic_reseau.shape

traffic_reseau.info()

traffic_reseau['src'].unique()

traffic_reseau['dst'].unique()

traffic_reseau['Protocol'].unique()

traffic_reseau.columns

traffic_reseau.isnull().sum()

traffic_reseau['rx_kbps'].fillna(traffic_reseau['rx_kbps'].mean(),
inplace=True)
traffic_reseau['tot_kbps'].fillna(traffic_reseau['tot_kbps'].mean(),
inplace=True)

traffic_reseau.isnull().sum()

# traffic_reseau['tot_kbps'].fillna(traffic_reseau['tot_kbps'].mean(),
inplace=True)

traffic_reseau['pktcount'].value_counts()

traffic_reseau['src'].value_counts()

traffic_reseau['dst'].value_counts()

traffic_reseau
```

```
traffic_reseau['Protocol'].unique()

# Mapping Protocol
Protocole_mapping = {
    'UDP':1,
    'TCP':2,
    'ICMP':3
}

traffic_reseau['Protocol'] =
traffic_reseau['Protocol'].map(Protocole_mapping)

traffic_reseau.head()

traffic_reseau['Protocol'].unique()

traffic_reseau.info()

traffic_reseau['src'].unique()

# IP Source Mapping
IP_source_mapping = {
    '10.0.0.1':1,
    '10.0.0.2':2,
    '10.0.0.4':4,
    '10.0.0.10':10,
    '10.0.0.3':3,
       '10.0.0.5':5,
    '10.0.0.13':13,
    '10.0.0.6':6,
    '10.0.0.20':20,
    '10.0.0.11':11,
       '10.0.0.12':12,
    '10.0.0.18':18,
    '10.0.0.8':8,
    '10.0.0.7':7,
    '10.0.0.9':9,
       '10.0.0.14':14,
    '10.0.0.15':15,
    '10.0.0.16':16,
    '10.0.0.17':17
}

traffic_reseau['src'] = traffic_reseau['src'].map(IP_source_mapping)

traffic_reseau.head()

traffic_reseau['dst'].unique()

# IP Dest Mapping
IP_Dst_Mapping = {
```

```
    '10.0.0.8':8,
    '10.0.0.7':7,
    '10.0.0.3':3,
    '10.0.0.5':5,
    '10.0.0.10':10,
       '10.0.0.13':13,
    '10.0.0.1':1,
    '10.0.0.11':11,
    '10.0.0.2':2,
    '10.0.0.4':4,
       '10.0.0.9':9,
    '10.0.0.6':6,
    '10.0.0.14':14,
    '10.0.0.15':15,
    '10.0.0.12':12,
       '10.0.0.16':16,
    '10.0.0.17':17,
    '10.0.0.18':18
}

traffic_reseau['dst'] = traffic_reseau['dst'].map(IP_Dst_Mapping)

traffic_reseau.head()
```

3. Separation de donnees

```
# Separation de donnees

traffic_reseau.loc[1:2]

traffic_reseau.head()

traffic_reseau.iloc[1:4, 2:4]

X = traffic_reseau.iloc[ :, :-1]

X

y = traffic_reseau.iloc[:,-1]

y

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

X_train.shape

print("X train:", X_train.shape)
print("y train:", y_train.shape)
```

```
print("X test:", X_test.shape)
print("y test:", y_test.shape)
```

4. Creation des Algorithmes

4.1 Logistic Regression

```
from sklearn.linear_model import LogisticRegression

# Creation d'une variable
lr_model = LogisticRegression()

# Entrainement
lr_model.fit(X_train, y_train)

y_pred_logistic = lr_model.predict(X_test)

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

confusion_matrix(y_test, y_pred_logistic)

ConfusionMatrixDisplay.from_estimator(lr_model, X_test, y_test)

from sklearn.metrics import accuracy_score, precision_score,
recall_score, mean_squared_error


print("Accuracy de Logistic Regression:", accuracy_score(y_test,
y_pred_logistic) * 100)
```

4.2 Random Forest

```
random_forest_model = RandomForestClassifier()

random_forest_model.fit(X_train, y_train)

y_pred_rf = random_forest_model.predict(X_test)

y_pred_rf

confusion_matrix(y_test, y_pred_rf)

accuracy_score(y_test, y_pred_rf)

X

pd.set_option('display.max_columns', None)
```

```
X
```

Simple prediction

```
random_forest_model.predict([[11425,  1,    1,    8,    45304,
     48294064,  100, 716000000, 1.010000e+11,   3,    1943, 13535,
     14428310,  451,  0,    1,    3,    143928631, 3917, 0,    0.0,  0.0
]])
```