Network Traffic Classification using KNN, Naive Bayes and SVM

KNN MODEL:

#Import all required libraries.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score,precision_score,classification_report
```

#Load dataset and understand the data.

```
data=pd.read_csv("nw.csv")
data.info()
```

#Separate features and targets.

```
x=data.drop("class",axis=1)
y=data["class"]
```

#Split the dataset in to training set and test set.

```
x_train,x_test, y_train, y_test = train_test_split( x, y, test_size=0.3, random_state=42,
stratify=y )
```

#Apply standard scaler.

```
scaler=StandardScaler()
x_train_scaled=scaler.fit_transform(x_train)
x_test_scaled=scaler.fit_transform(x_test)
```

#Train the model using KNN

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train_scaled, y_train)
y_pred_knn = knn.predict(x_test_scaled)
```

#Evaluate the performance

```
acc_knn=accuracy_score(y_test, y_pred_knn)
print("KNN Accuracy:",acc_knn)
print("Classification Report for KNN:")
```

c_knn=classification_report(y_test, y_pred_knn)

print(c_knn)

```
KNN Accuracy: 0.9866666666666667
Classification Report for KNN:
              precision    recall  f1-score   support

  bruteForce       1.00      1.00      1.00        60
   httpFlood       1.00      0.99      1.00       172
   icmp-echo       0.99      0.95      0.97       190
      normal       0.96      0.98      0.97       180
   slowloris       0.97      1.00      0.99       234
    slowpost       0.99      0.99      0.99       144
     tcp-syn       0.99      0.99      0.99       288
   udp-flood       1.00      0.99      0.99       232

    accuracy                           0.99      1500
   macro avg       0.99      0.99      0.99      1500
weighted avg       0.99      0.99      0.99      1500
```

NAVES BAYE'S MODEL:

#Import all required libraries.

import pandas as pd

import numpy as np

import seaborn as sns

from sklearn.naive_bayes import GaussianNB

from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

#Load the dataset and create dataframe.

df = pd.read_csv("nw.csv")

#**Separate features and targets (Encode categorical value**

X = df.drop(columns=["class"])

y = df["class"].astype("category").cat.codes

#Split the dataset into train set and test set.

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

#Normalize features.

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)

#Train the model with Naïve Bayes Classifier

nb = GaussianNB()

nb.fit(X_train_scaled, y_train)

y_pred_nb = nb.predict(X_test_scaled)

 # Evaluate performance and display the results.

accuracy_nb = accuracy_score(y_test, y_pred_nb)

report_nb = classification_report(y_test, y_pred_nb, zero_division=1)

conf_matrix = confusion_matrix(y_test, y_pred_nb)

print(f"Naïve Bayes Accuracy: {accuracy_nb:.4f}")

print("Classification Report:\n", report_nb)

```
Naïve Bayes Accuracy: 0.6550
Classification Report:
              precision    recall  f1-score   support

           0       0.36      1.00      0.53        40
           1       1.00      0.92      0.96       115
           2       0.54      0.64      0.58       126
           3       0.75      0.64      0.69       120
           4       0.56      1.00      0.72       156
           5       0.64      0.99      0.78        96
           6       0.99      0.38      0.54       192
           7       1.00      0.18      0.31       155

    accuracy                           0.66      1000
   macro avg       0.73      0.72      0.64      1000
weighted avg       0.78      0.66      0.63      1000
```

SVM MODEL:

#import required libraries

```
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
# Load dataset
df = pd.read_csv("nw.csv")
# Separate features and target variable and encode categorical target
X = df.drop(columns=["class"])
y = df["class"].astype("category").cat.codes
# Split dataset
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=42,
stratify=y)
# Normalize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
# Apply SVM Model # Radial Basis Function (RBF) kernel
svm_model = SVC(kernel="rbf", C=1.0, gamma="scale", random_state=42)
svm_model.fit(X_train_scaled, y_train)
y_pred_svm = svm_model.predict(X_test_scaled)
# Evaluate Performance
accuracy_svm = accuracy_score(y_test, y_pred_svm)
report_svm = classification_report(y_test, y_pred_svm, zero_division=1)
# Print results
print(f"SVM Accuracy: {accuracy_svm:.4f}")
print("SVM Classification Report:\n", report_svm)
```

```
SVM Accuracy: 0.9830
SVM Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        40
           1       1.00      0.96      0.98       115
           2       0.98      0.94      0.96       126
           3       0.97      0.99      0.98       120
           4       0.99      1.00      1.00       156
           5       1.00      0.99      0.99        96
           6       0.95      1.00      0.97       192
           7       1.00      1.00      1.00       155

    accuracy                           0.98      1000
   macro avg       0.99      0.98      0.98      1000
weighted avg       0.98      0.98      0.98      1000
```