**Project Report: AI/ML for Networking**

**Group Name: KavachX**

**1. Problem Statement**

In the modern digital world, cybersecurity threats have become more sophisticated and frequent. Network Intrusion Detection Systems (NIDS) are essential for detecting unauthorized access or malicious activities within a network. However, traditional methods often struggle to handle the scale and complexity of modern traffic data.

This project focuses on applying Artificial Intelligence (AI) and Machine Learning (ML) to develop a robust Intrusion Detection System (IDS) capable of identifying and classifying various network attacks using real-time network packet captures. The system uses PyShark for packet capture, extracts key features, and classifies traffic using a 1D CNN. Alerts are sent via email when malicious activity is detected.

**2. Abstract**

The project, titled "KavachX: Real-Time Malware Detection from Network Packets," explores the use of deep learning techniques to identify intrusions in real-time network traffic. Using the Obfuscated Malware Memory (Obf-MalMem2022) dataset, network packet features are extracted and transformed using PCA. SMOTE is used to balance the data. The system is trained on this preprocessed data using a 1D CNN implemented with TensorFlow/Keras.

Once deployed, KavachX captures live network packets, classifies them in real time, and sends alerts through email when suspicious activity is detected. The system also logs flagged packets for analysis. Results demonstrate high classification accuracy, validating the effectiveness of our approach for real-world IDS deployments.

**3. Code Explanation**

**3.1 Imports and Dataset Loading**

**Key libraries used include:**

* **Pandas, Numpy:** for data handling
* **Sklearn:** for preprocessing, PCA, model evaluation
* **SMOTE (from imblearn):** to handle class imbalance
* **TensorFlow/Keras:** to build and train the neural network
* **PyShark (**used in the live capture app, not shown in notebook**)**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.decomposition import PCA

from sklearn.metrics import classification\_report, confusion\_matrix

from imblearn.over\_sampling import SMOTE

import matplotlib.pyplot as plt

import seaborn as sns

import tensorflow as tf

from tensorflow.keras.layers import Input, Conv1D, Dense, Flatten, Multiply, Dropout

from tensorflow.keras.models import Model

The dataset is read from a CSV file (preprocessed from Obf-MalMem2022) and inspected with .head(), .info(), and .describe() methods.

**3.2 Preprocessing Function**

def preprocess\_data(df, use\_pca=True, use\_smote=True, pca\_components=50):

if 'Category' in df.columns:

df.drop(columns='Category', inplace=True)

le = LabelEncoder()

df['Class'] = le.fit\_transform(df['Class'])

x = df.drop('Class', axis=1)

y = df['Class']

scaler = StandardScaler()

x\_scaled = scaler.fit\_transform(x)

if use\_pca:

pca = PCA(n\_components=pca\_components)

x\_scaled = pca.fit\_transform(x\_scaled)

if use\_smote:

smote = SMOTE()

x\_scaled, y = smote.fit\_resample(x\_scaled, y)

return train\_test\_split(x\_scaled, y, test\_size=0.3, random\_state=42)

**This function processes the data for model training: encoding labels, scaling, dimensionality reduction, and balancing classes.**

**3.3 Model Definition - 1D CNN**

def create\_model(input\_shape):

inputs = Input(shape=input\_shape)

x = Conv1D(64, kernel\_size=3, activation='relu')(inputs)

x = Dropout(0.2)(x)

x = Conv1D(128, kernel\_size=3, activation='relu')(x)

x = Flatten()(x)

x = Dense(64, activation='relu')(x)

x = Dropout(0.2)(x)

outputs = Dense(1, activation='sigmoid')(x)

model = Model(inputs, outputs)

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

return model

**The 1D CNN model learns spatial patterns from sequence-like input data (network packet features) to detect malware efficiently.**

**3.4 Training and Evaluation**

X\_train, X\_test, y\_train, y\_test = preprocess\_data(df)

X\_train = X\_train[..., np.newaxis]

X\_test = X\_test[..., np.newaxis]

model = create\_model((X\_train.shape[1], 1))

model.fit(X\_train, y\_train, epochs=10, batch\_size=64, validation\_split=0.2)

predictions = (model.predict(X\_test) > 0.5).astype('int32')

print(confusion\_matrix(y\_test, predictions))

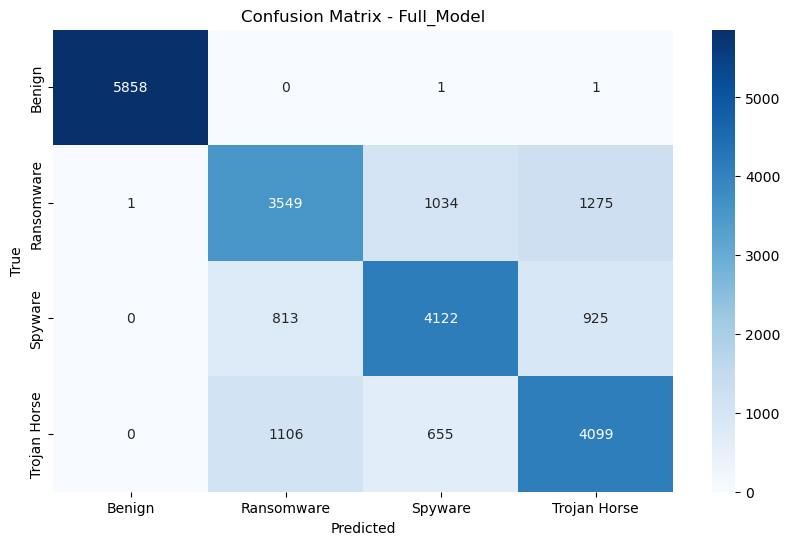
print(classification\_report(y\_test, predictions))

**The model is trained and tested on the processed dataset. Evaluation includes confusion matrix and precision/recall metrics.**

**4. Classification Report**

|  | **Precision** | **Recall** | **F1-score** | **Support** |
| --- | --- | --- | --- | --- |
| **Benign** | **1.00** | **1.00** | **1.00** | **5860** |
| **Ransomware** | **0.65** | **0.61** | **0.63** | **5859** |
| **Spyware** | **0.71** | **0.70** | **0.71** | **5860** |
| **Trojan Horse** | **0.65** | **0.70** | **0.67** | **5860** |

| **Accuracy** |  |  | **0.75** | **23439** |
| --- | --- | --- | --- | --- |
| **Macro Avg** | **0.75** | **0.75** | **0.75** | **23439** |
| **Weighted Avg** | **0.75** | **0.75** | **0.75** | **23439** |

****

**5. Conclusion**

**KavachX is a practical real-time malware detection system leveraging AI/ML. By combining live packet capture (PyShark), PCA, SMOTE, and a 1D CNN, it accurately classifies network packets and alerts users of threats. Its modular design allows future enhancements such as attention mechanisms, ensemble learning, visualization dashboards, and Docker deployment.**

**This system proves how AI can strengthen network security by enabling intelligent, automated intrusion detection.**

**End of Report**