About the Intelligent Intrusion Detection System (IDS) Model

This project implements a deep learning-based **Intrusion Detection System (IDS)** using a combination of **LightGBM, LSTM, and Deep Neural Networks (DNN)**. The goal is to classify network traffic as either **benign or malicious** using the **UNSW-NB15 dataset**. The system performs both **binary classification (normal vs. attack)** and **multiclass classification (attack type detection)** to enhance cybersecurity defenses.

Key Features of the IDS Model:

1. **Dataset Preprocessing**
    - Loads and processes the **UNSW-NB15 dataset**.
    - Drops unnecessary columns (like "id") and encodes categorical features.
    - Scales numerical features using **RobustScaler** to handle outliers.
2. **Feature Selection with LightGBM and SHAP**
    - Uses **LightGBM** to train an initial classifier.
    - Leverages **SHAP (SHapley Additive exPlanations)** to identify the **top 15 most important features**.
    - Reduces the dataset to these important features for efficient training.
3. **Handling Class Imbalance with SMOTE**
    - Uses **Synthetic Minority Over-sampling Technique (SMOTE)** to balance the dataset.
    - Ensures that both binary and multiclass classification models get balanced data.

Multiclass Classification (LSTM with Attention)

- Uses a **Bidirectional LSTM (Long Short-Term Memory)** network to capture sequential dependencies in network traffic.
- Incorporates **Batch Normalization** and **Dropout** to improve generalization.
- Implements **Optuna for hyperparameter tuning**, optimizing:
    - LSTM units
    - Dropout rates

    - Learning rate
    - Batch size
- Trains with **class weights** to counteract class imbalance.
- Saves the trained model as **multiclass_model.h5**.

Binary Classification (Deep Neural Network - DNN)

- Uses a **Deep Neural Network (DNN)** with:
    - **SELU activation** for better learning stability.
    - **Batch Normalization** and **Dropout layers** for regularization.
- Compiles with **AdamW optimizer** and **Binary Crossentropy loss**.
- Trains for 20 epochs and evaluates accuracy.
- Saves the trained model as **binary_model.h5**.

Model Performance & Evaluation

- The multiclass model is evaluated on test data to compute accuracy.
- The binary classification model is tested separately for accuracy.
- Both models are saved for deployment in cybersecurity applications.

Conclusion

This project provides a robust **Intrusion Detection System** using deep learning, enabling both binary and multiclass classification of network traffic. With **feature selection, class balancing, and hyperparameter tuning**, the models achieve high accuracy, making them suitable for real-world cybersecurity applications.

Screenshots

```
[LightGBM] [Info] Number of positive: 119341, number of negative: 56000
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of tes
ting was 0.009417 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 6062
[LightGBM] [Info] Number of data points in the train set: 175341, number of u
sed features: 42
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.680622 -> initscore=0.75663
3
[LightGBM] [Info] Start training from score 0.756633
Selected Features after LightGBM + SHAP:
Index(['sloss', 'rate', 'dmean', 'sinpkt', 'ct_dst_src_ltm', 'service',
       'ct_srv_src', 'smean', 'dbytes', 'ct_state_ttl', 'ct_srv_dst', 'sbyte
s',
       'proto', 'ct_dst_sport_ltm', 'sttl'],
      dtype='object')
```


```
[LightGBM] [Info] Number of positive: 119341, number of negative: 56000
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of tes
ting was 0.009417 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 6062
[LightGBM] [Info] Number of data points in the train set: 175341, number of u
sed features: 42
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.680622 -> initscore=0.75663
3
[LightGBM] [Info] Start training from score 0.756633
Selected Features after LightGBM + SHAP:
Index(['sloss', 'rate', 'dmean', 'sinpkt', 'ct_dst_src_ltm', 'service',
       'ct_srv_src', 'smean', 'dbytes', 'ct_state_ttl', 'ct_srv_dst', 'sbyte
s',
       'proto', 'ct_dst_sport_ltm', 'sttl'],
      dtype='object')
```

```
[I 2025-02-13 02:43:47,289] A new study created in memory with name: no-name-
3dbb46d1-ff75-4f40-879c-1ab0758924b7
C:\Users\LENOVO FLEX\AppData\Local\Temp\ipykernel_17816\2625913597.py:89: Fut
ureWarning: suggest_loguniform has been deprecated in v3.0.0. This feature wi
ll be removed in v6.0.0. See https://github.com/optuna/optuna/releases/tag/v
3.0.0. Use suggest_float(..., log=True) instead.
  optimizer = Adam(learning_rate=trial.suggest_loguniform("lr", 1e-4, 1e-2))
[I 2025-02-13 02:48:25,229] Trial 0 finished with value: 0.752773106098175 an
d parameters: {'units_1': 224, 'dropout_1': 0.49679924891096944, 'units_2': 4
8, 'dropout_2': 0.36168270361383803, 'lr': 0.003085763610625092, 'batch_siz
e': 128}. Best is trial 0 with value: 0.752773106098175.
C:\Users\LENOVO FLEX\AppData\Local\Temp\ipykernel_17816\2625913597.py:89: Fut
ureWarning: suggest_loguniform has been deprecated in v3.0.0. This feature wi
ll be removed in v6.0.0. See https://github.com/optuna/optuna/releases/tag/v
3.0.0. Use suggest_float(..., log=True) instead.
  optimizer = Adam(learning_rate=trial.suggest_loguniform("lr", 1e-4, 1e-2))
```

```
C:\Users\LENOVO FLEX\AppData\Local\Temp\ipykernel_17816\2625913597.py:89: Fu
ureWarning: suggest_loguniform has been deprecated in v3.0.0. This feature w
ll be removed in v6.0.0. See https://github.com/optuna/optuna/releases/tag/v
3.0.0. Use suggest_float(..., log=True) instead.
  optimizer = Adam(learning_rate=trial.suggest_loguniform("lr", 1e-4, 1e-2))
[I 2025-02-13 02:59:19,828] Trial 2 finished with value: 0.7517750859260559
nd parameters: {'units_1': 160, 'dropout_1': 0.34788595053843974, 'units_2':
96, 'dropout_2': 0.42364372329665734, 'lr': 0.001067123054310313, 'batch_siz
e': 64}. Best is trial 0 with value: 0.752773106098175.
C:\Users\LENOVO FLEX\AppData\Local\Temp\ipykernel_17816\2625913597.py:89: Fu
ureWarning: suggest_loguniform has been deprecated in v3.0.0. This feature w
ll be removed in v6.0.0. See https://github.com/optuna/optuna/releases/tag/v
3.0.0. Use suggest_float(..., log=True) instead.
  optimizer = Adam(learning_rate=trial.suggest_loguniform("lr", 1e-4, 1e-2))
[I 2025-02-13 03:15:14,908] Trial 3 finished with value: 0.7461575865745544
```

```
Epoch 30/50
7000/7000 ━━━━━━━━━━━━━━━━━━━━ 23s 3ms/step - accuracy: 0.7202 - loss: 0.6978
- val_accuracy: 0.7569 - val_loss: 0.5964
Epoch 31/50
7000/7000 ━━━━━━━━━━━━━━━━━━━━ 41s 3ms/step - accuracy: 0.7197 - loss: 0.7000
- val_accuracy: 0.7624 - val_loss: 0.5934
Epoch 32/50
7000/7000 ━━━━━━━━━━━━━━━━━━━━ 23s 3ms/step - accuracy: 0.7202 - loss: 0.6972
- val_accuracy: 0.7397 - val_loss: 0.6284
1096/1096 ━━━━━━━━━━━━━━━━━━━━ 2s 1ms/step - accuracy: 0.7392 - loss: 0.6303

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or
`keras.saving.save_model(model)`. This file format is considered legacy. We r
ecommend using instead the native Keras format, e.g. `model.save('my_model.ke
ras')` or `keras.saving.save_model(model, 'my_model.keras')`.

Multiclass Model Accuracy: 73.97%
Multiclass Model saved successfully.
```

```
Epoch 18/20
5967/5967 ━━━━━━━━━━━━━━━━━━━━ 11s 2ms/step - accuracy: 0.9313 - loss: 0.1430
- val_accuracy: 0.9145 - val_loss: 0.1946
Epoch 19/20
5967/5967 ━━━━━━━━━━━━━━━━━━━━ 11s 2ms/step - accuracy: 0.9313 - loss: 0.1419
- val_accuracy: 0.9301 - val_loss: 0.1409
Epoch 20/20
5967/5967 ━━━━━━━━━━━━━━━━━━━━ 11s 2ms/step - accuracy: 0.9324 - loss: 0.1406
- val_accuracy: 0.9291 - val_loss: 0.1415
1096/1096 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step - accuracy: 0.9283 - loss: 0.1421

WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or
`keras.saving.save_model(model)`. This file format is considered legacy. We r
ecommend using instead the native Keras format, e.g. `model.save('my_model.ke
ras')` or `keras.saving.save_model(model, 'my_model.keras')`.

Binary Model Accuracy: 92.91%
Binary Model saved successfully.
```

```
WARNING:tensorflow:5 out of the last 5 calls to <function TensorFlowTrainer.make
_predict_function.<locals>.one_step_on_data_distributed at 0x0000026755525F80> t
riggered tf.function retracing. Tracing is expensive and the excessive number of
tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) pas
sing tensors with different shapes, (3) passing Python objects instead of tensor
s. For (1), please define your @tf.function outside of the loop. For (2), @tf.fu
nction has reduce_retracing=True option that can avoid unnecessary retracing. Fo
r (3), please refer to https://www.tensorflow.org/guide/function#controlling_ret
racing and https://www.tensorflow.org/api_docs/python/tf/function for  more deta
ils.
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step
Binary Classification Result:
Prediction: Attack

Multiclass Classification Result:
Predicted Attack Type: Fuzzers
```