Reflection on Class Imbalance in NSL-KDD

From your results, you'll notice that:

- **Normal traffic** and **DoS attacks** dominate the dataset.
- **Probe attacks** are fewer but still sizable.
- **R2L** (Remote-to-Local) and **U2R** (User-to-Root) attacks are **very rare**, often contributing less than 1% of the total samples.

This imbalance means:

- A naïve model might be biased toward the majority classes (Normal & DoS).
- Minority attacks (especially **U2R** and **R2L**) may be overlooked — but these are the most *critical* since they represent high-severity intrusions.
- Accuracy as a metric can be misleading. For example, a model predicting only Normal/DoS might still achieve >90% accuracy but fail at detecting U2R/R2L.

⚖️ Mitigation Strategies

1. Resampling Techniques

- **Oversampling minority classes**
    - Use **SMOTE (Synthetic Minority Over-sampling Technique)** or **ADASYN** to generate synthetic samples of R2L and U2R.
- **Undersampling majority classes**
    - Reduce the number of Normal/DoS records to balance proportions.
- **Hybrid approaches**
    - Combine undersampling with SMOTE for better balance.

---

2. Class Weighting

- Many ML models (e.g., Logistic Regression, SVM, Random Forest, XGBoost, Neural Nets in PyTorch/TF) allow assigning **higher weights** to minority classes.
- Example: class_weight='balanced' in scikit-learn.
- This penalizes the model more for misclassifying rare attack categories.

---

3. Data Augmentation

- Beyond SMOTE, you could **generate synthetic variations** of rare attacks using:
    - Noise injection,
    - Feature perturbation,

- GAN-based augmentation (advanced option).

---

4. Ensemble Methods

- Use models like **Balanced Random Forest** or **EasyEnsemble**.
- These are designed to handle imbalance by sampling subsets of the data.

---

5. Evaluation Metrics

- Don't rely solely on accuracy.
- Use:
  - **Precision, Recall, F1-score** (macro-averaged to treat all classes equally),
  - **Confusion matrix** (to see per-class performance),
  - **ROC-AUC per class**.

---

6. Stratified Splitting

- Ensure training/test sets preserve class ratios (especially important for minority classes).
- Use StratifiedKFold for cross-validation.