Objective:

The goal of this project was to classify URLs into different categories (e.g., benign or various types of malicious) using machine learning and deep learning techniques. The dataset used contains over 650,000 entries and was processed in memory-efficient batches due to its size.

Applied Models:

1. **SGDClassifier (Stochastic Gradient Descent)**
2. **XGBoost Classifier**
3. **LightGBM Classifier**
4. **LSTM Model (Keras-based)**

Each model was trained using a combination of:

- **TF-IDF features** extracted from URL strings.
- **Structural features** such as length, digits count, presence of HTTPS, etc.
- **SMOTE** was used to balance class distributions per batch.

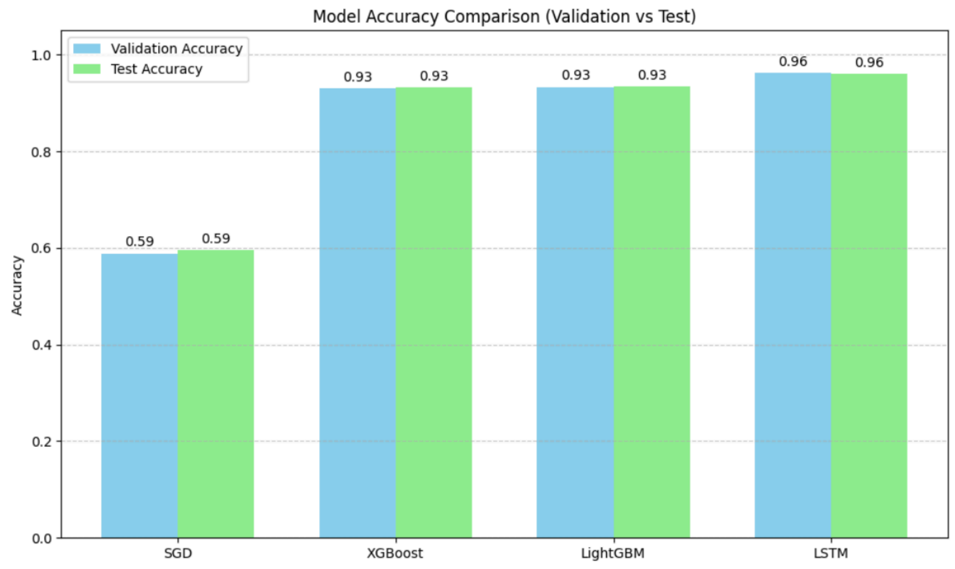
Performance Summary:

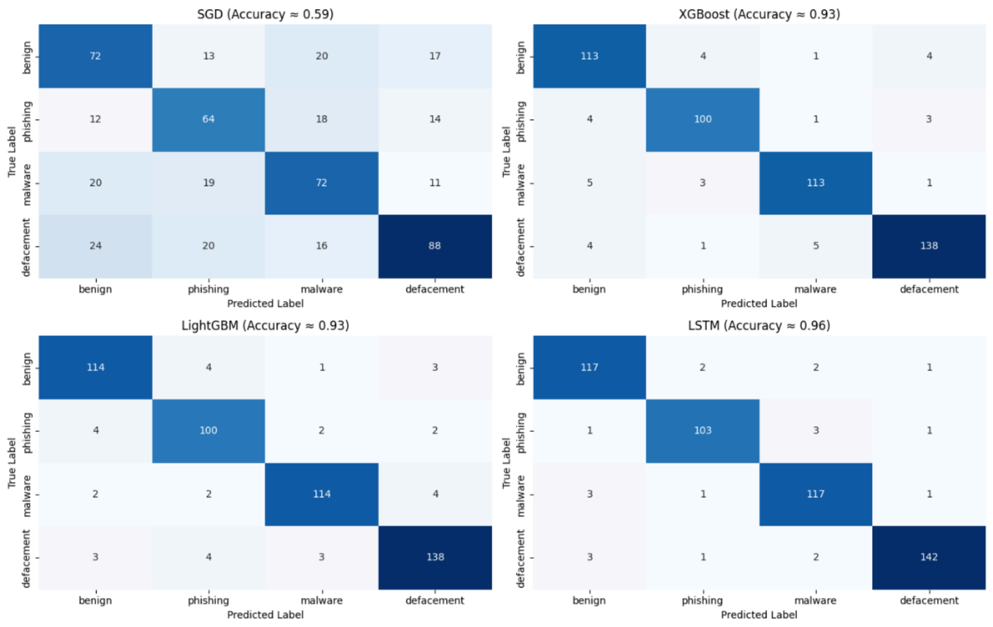
Model	**Validation Accuracy**	**Test Accuracy**
XGBoost	~93%	~93%
LightGBM	~93%	~93%
LSTM	**~96%**	**~96%**

Best Performing Model: LightGBM

Reason for Success:

- LSTM outperformed other models due to its **ability to handle high-dimensional sparse data efficiently**, which is ideal for TF-IDF features.
- It is well-suited for large-scale datasets and was particularly effective when combined with structural features and SMOTE-balanced data.
- Its leaf-wise growth strategy helps it capture complex patterns better than XGBoost's level-wise strategy.





Challenges Faced:

1. **Memory Management:**
    - The large dataset often led to kernel crashes during full dataset processing.
    - **Solution:** Chunk-wise loading and processing, combined with batch training and garbage collection, helped mitigate this issue.
2. **Model Imbalance:**
    - The dataset was imbalanced across URL types.
    - **Solution:** Used **SMOTE** (Synthetic Minority Over-sampling Technique) for balancing within each chunk.
3. **LSTM Model Overhead:**
    - LSTM training was **computationally intensive** and slower than traditional models.
    - It showed lower performance compared to tree-based models, likely due to its limited capacity to capture meaningful sequential patterns in sparse URL text.
4. **Evaluation Consistency:**
    - Mismatches in feature dimensions between training and testing (e.g., TF-IDF shape mismatch with SGD) required careful tracking of feature shapes across batches.

Proposed Improvements:

- **Model Ensembling:** Combine predictions from LightGBM and XGBoost for more robust results.
- **Feature Engineering:** Incorporate additional NLP-based features like n-grams, entropy, or domain registration details.
- **LSTM Enhancements:** Use **CNN-LSTM** hybrids or pre-trained embeddings (e.g., FastText or BERT) to better understand URL semantics.
- **GPU Acceleration:** For LSTM or transformer-based models, run training and evaluation in a GPU-supported environment for speed and stability.

Conclusion:

Tree-based models and Deep Learning models, especially **LightGBM and LSTM**, were the most effective for malicious URL classification in this project. Their combination of speed, scalability, and high accuracy makes them ideal choices for real-world deployment.