Introduction

This report provides an in-depth analysis of the machine learning models applied to classify malicious URLs. The primary focus is on evaluating model performance, identifying key challenges, and proposing improvements for future iterations.

Model Performance Comparison

Three key approaches were used to train and evaluate models:

1. **Random Forest Classifier on Numerical Data**
2. **LLM for Processing**
3. **LSTM Model for URL Sequence Classification**

Random Forest Classifier

The Random Forest model was trained using numerical features extracted from the dataset. The dataset was processed by selecting only numerical columns and handling class imbalance through **SMOTE** and **RandomOverSampler**.

- **Training Process**:
  - **Feature Selection**: Only numerical columns were used.
  - **Data Balancing**: SMOTE and RandomOverSampler were applied to balance class distributions.
  - **Hyperparameters**: The model was trained with 150 trees, max depth of 12, and class weight adjustment.
- **Evaluation Metrics**:
  - **Accuracy**: The model achieved an accuracy of approximately 99%.
  - **Classification Report**: Precision, recall, and F1-score were 1.0 indicating effective classification.
  - **Confusion Matrix**: Most misclassifications occurred in borderline cases.

```
Random Forest Accuracy: 0.999961717505819
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85616
           1       1.00      1.00      1.00     19062
           2       1.00      1.00      1.00      4729
           3       1.00      1.00      1.00     18817
           4       1.00      1.00      1.00      2384

    accuracy                           1.00    130608
   macro avg       1.00      1.00      1.00    130608
weighted avg       1.00      1.00      1.00    130608
```

LLM for Classification

2.1 GPT-2 (lvwerra/gpt2-imdb)

- **Model Chosen:** A GPT-2 model fine-tuned for sentiment classification.
- **Implementation:** Used Hugging Face's AutoTokenizer and AutoModelForSequenceClassification to fine-tune on URL classification.
- **Issue:** The model requires substantial computational power. The fine-tuning process failed due to **GPU memory constraints**.

2.2 DistilGPT-2

- **Model Chosen:** A lighter version of GPT-2 to reduce computational demands.
- **Implementation:** Similar to GPT-2, adjusted training batch sizes and used stratified train-test splits.
- **Issue:** Even with reduced model size, training still encountered **out-of-memory (OOM) errors**, preventing successful model convergence.

2.3 BERT

- **Model Chosen:** bert-base-uncased for masked language modeling and classification.
- **Implementation:** Attempted classification by fine-tuning the last few layers.
- **Issue:** BERT struggled with URL-based data, as **URLs lack meaningful word embeddings**, making classification inefficient. Additionally, OOM errors persisted during training.

2.4 BART

- **Model Chosen:** facebook/bart-base, a seq2seq transformer designed for NLP tasks.
- **Implementation:** Used for classification in a similar fashion to BERT.
- **Issue:** Like BERT, BART failed to handle URL-based data effectively, and fine-tuning required high computational power.

2.5: TabNet – A Transformer-Based Approach

- **Model Used**: TabNetClassifier
- **Performance**: Achieved **99% accuracy** on test data
- **Key Features**:
    - Employs **attention mechanisms** for feature selection
    - Optimized for tabular data processing
    - Supports **sequential decision-making** for better classification
- **Results**: The model demonstrated high robustness and accuracy, making it a strong candidate for structured dataset classification.
- **Next Steps**: Fine-tuning hyperparameters and incorporating additional features could further enhance its performance.

```
epoch 0  | loss: 0.07903 | validation_accuracy: 0.99995 |  0:00:23s
epoch 1  | loss: 0.00547 | validation_accuracy: 0.99969 |  0:00:47s
epoch 2  | loss: 0.00181 | validation_accuracy: 0.99963 |  0:01:10s
Stop training because you reached max_epochs = 3 with best_epoch = 0 and best_validation_accuracy =
```

LSTM Model for URL Classification

An LSTM-based deep learning model was trained on tokenized URL sequences to analyze malicious URLs.

- **Training Process**:
  - **Tokenization**: The URLs were converted into sequences of character-level tokens.
  - **Padding**: Sequences were padded to a fixed length of 200.
  - **Model Architecture**:
    - **Embedding Layer**: Converts tokens into dense vector representations.
    - **LSTM Layers**: Two stacked LSTM layers (64 and 32 units) capture sequential patterns.
    - **Dense Layers**: Fully connected layers with 0.5 dropout to reduce overfitting.
- **Evaluation Metrics**:
  - **Accuracy**: LSTM achieved an accuracy of approximately 97%, which is slightly below that of Random Forest.
  - **Training Challenges**: Overfitting was mitigated with dropout and balanced training data.

LSTM Performance:

	precision	recall	f1-score	support
benign	0.98	0.99	0.99	85757
defacement	0.99	1.00	0.99	18929
malware	0.99	0.94	0.96	4672
phishing	0.94	0.91	0.92	18924
spam	1.00	1.00	1.00	2326
accuracy			0.98	130608
macro avg	0.98	0.97	0.97	130608
weighted avg	0.98	0.98	0.98	130608

Challenges Faced

Several challenges were encountered during this analysis:

1. Data Transformation Complexity

The dataset required significant preprocessing. Raw data included URLs, categorical labels, and numerical features, making it necessary to carefully engineer features for ML models.

2. Handling Numerical Data for LLMs and LSTMs

Large Language Models and deep learning models like LSTM are optimized for textual data. **Numerical data cannot be directly used** in these models, requiring extensive transformation. For BERT, numerical values were converted into descriptive text. For LSTM, raw URLs were tokenized at the character level and Tablet showed approx 99% accuracy.

3. Model Selection for Different Data Types

- **Random Forest outperformed LSTM and Tablet** because numerical data is inherently better suited for traditional machine learning models.
- **LSTM worked well for URL-based classification**, but required complex preprocessing.
- **LLM struggled with numerical transformations**, making it less effective in this specific case and in my case it kind of did not work well.

Proposed Improvements

To enhance performance and improve future classification attempts, the following strategies are recommended:

1. **Hybrid Model Approach**:
    - Use **Random Forest** for numerical data processing.
    - Apply **LLMs** to classify textual components such as URL patterns and metadata.
    - Combine predictions from both models to improve overall accuracy.
2. **Feature Engineering Enhancements**:
    - Extract additional features from URLs (e.g., domain age, presence of keywords).
    - Apply PCA (Principal Component Analysis) to improve numerical feature representation.
3. **Alternative ML Models for Numerical Data**:
    - Experiment with **Gradient Boosting Models** (e.g., XGBoost, LightGBM) for potentially better results.
    - Tune hyperparameters to further optimize classification performance.

Conclusion

The analysis demonstrated that **Random Forest performed better than LSTM and LLM-based classification** due to the numerical nature of the dataset. However, **LSTMs were effective for URL-based classification**, while **LLM struggled with numerical transformations** but **Tablet** worked well. Future improvements should explore hybrid approaches that integrate all three methodologies for optimal classification performance.