CS658 - Project - Presentation 2024

Malicious URL Classification Using Machine Learning

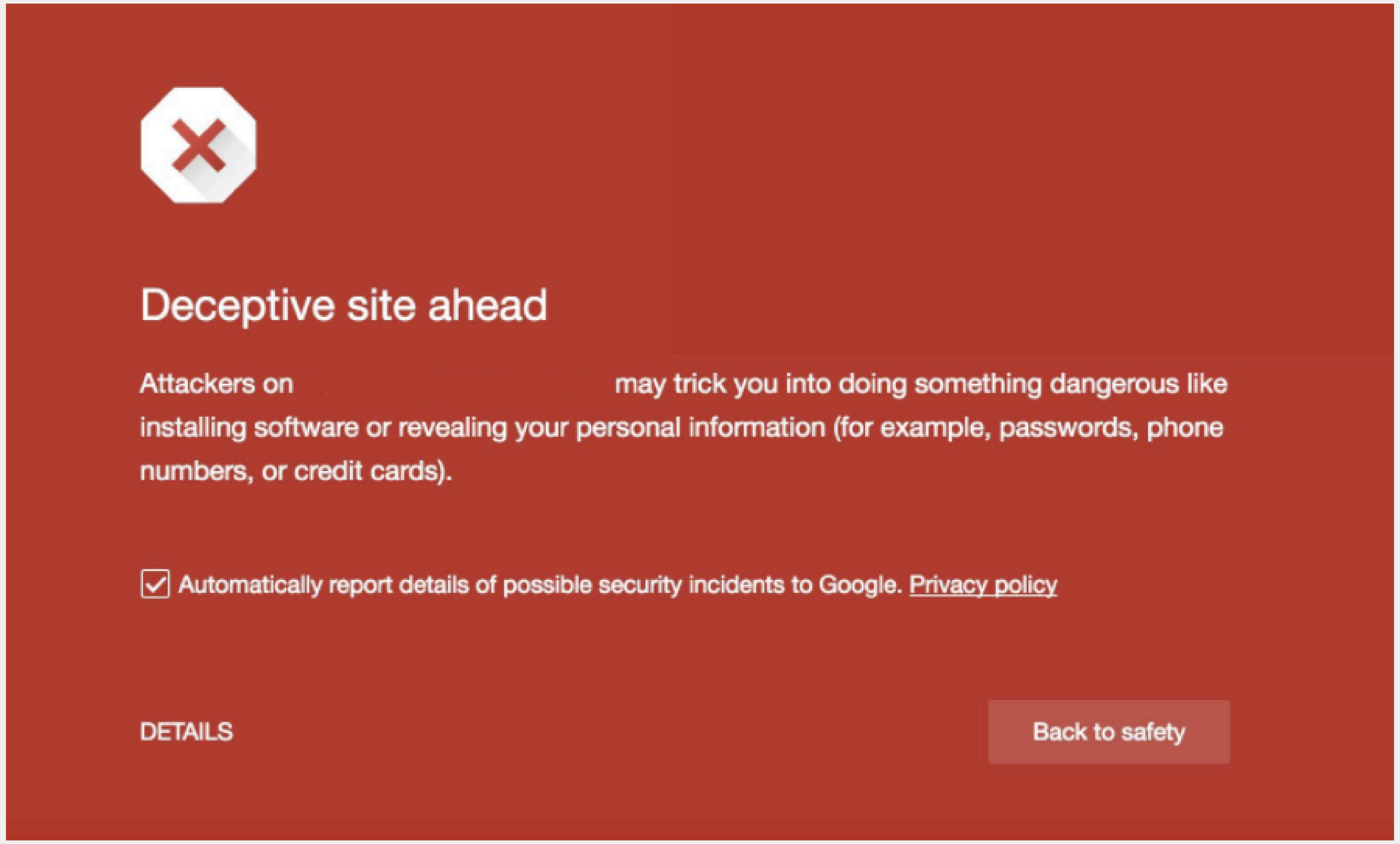
Presented by Group 2:-

Raj Vinayak **Gottupulla Venkata Aman**

Ashutosh Agarwal **Varun Tandon**

Imagine clicking on a harmless-looking link from a friend, expecting it to be a cool meme, a song recommendation, or just something interesting. But instead… Never Gonna Give You Up starts blasting from your speakers. Congratulations, you've been Rickrolled!

"Harmless, right? But what if that link wasn’t just a Rickroll? What if it led to something far worse?"



Deceptive site ahead

Attackers on may trick you into doing something dangerous like installing software or revealing your personal information (for example, passwords, phone numbers, or credit cards).

☑ Automatically report details of possible security incidents to Google. Privacy policy

Objective of the Project:

Problem: Detecting Malicious URLs

Types of Attacks Addressed: Our project focuses on detecting harmful web addresses (URLs) which often act as conduits for phishing, malware distribution, defacement, and spam attacks.

Cybersecurity Context:

URLs have become a common attack vector for cybercriminals, often disguised to appear legitimate to mislead users. This can lead to stolen credentials, malware downloads, and compromised systems.

Importance of Detection:

Efficient malicious URL detection prevents unauthorized access, data breaches, and malware infiltration, which are costly and time-consuming to resolve.

Goal:

Develop a machine learning model to classify URLs as malicious or benign and, among malicious URLs, further categorize them into phishing, spam, defacement, etc.

Advantages of Machine Learning for URL Classification

Data Processing Efficiency

Machine learning algorithms can efficiently process and analyze vast amounts of URL data, identifying complex patterns and anomalies that signify malicious activity. This capability is essential for keeping pace with the rapid increase in URL generation and the evolving tactics of cybercriminals.

Dynamic Adaptability

The adaptability of machine learning models allows them to continuously learn from new data and feedback, improving their accuracy over time. This ensures that they can effectively recognize emerging threats and adjust to new malicious tactics, significantly enhancing cybersecurity defenses.

Dataset

Dataset Source

The Canadian Institute for Cybersecurity (CIC) provided this dataset, widely used in cybersecurity research.

Dataset Composition

This dataset contains key attributes of URLs such as entropy, URL path, directory names, filenames, and special characters to represent both structural and content-based aspects.

Data Collection Method by CIC

The CIC developed this dataset by monitoring network traffic and gathering URLs from different attack categories.

URLs were classified by analyzing attributes like structure, content, and behavior to ensure a comprehensive set of both malicious and benign URLs.

Reason for Choosing This Dataset

The dataset offers a diverse set of URL types, enabling robust training for a machine learning model.

CIC is a recognized research institution in cybersecurity, making this dataset a credible and reliable resource.

Methods and Data Preprocessing

01

Classification Methodology

We utilized a supervised machine learning approach for URL classification.

Selected Models: Decision Tree, Logistic Regression, and Random Forest for their effectiveness with structured data in cybersecurity applications.

02

Beyond Python Libraries

Anomaly Detection: Implemented Isolation Forest to identify unusual URL patterns.

Data Preprocessing Techniques:

Handled Missing Data: Replaced missing values with column means for data continuity.

Feature Correlation and Selection: Removed highly correlated features to enhance model efficiency.

Scaling: Applied MinMaxScaler to place features on a uniform scale (0-1) for unbiased model training.

03

Label Encoding

Encoded the target variable, URL_Type_obf_Type, to numerical values using LabelEncoder for machine learning algorithm processing.

04

Why These Methods?

These preprocessing steps ensure high-quality data preparation, reducing noise, preventing overfitting, and ensuring stable model performance across URL types.

Model Training and Hyperparameter Tuning

01

Initial Model Selection

Decision Tree, Logistic Regression, and Random Forest:

Decision Tree: Known for interpretability, it can easily handle feature relationships in a tree structure.

Logistic Regression: Acts as a robust baseline model for binary classification.

Random Forest: A highly effective ensemble method that can capture complex feature interactions and reduce overfitting compared to a single decision tree.

02

Hyperparameter Tuning

Grid Search with Cross-Validation:

We used GridSearchCV to systematically explore a range of hyperparameter combinations, using 5-fold cross-validation for reliable evaluation.

03

Scoring Metric

Accuracy was used as the primary metric, with additional considerations for F1-score to handle class imbalances effectively.

04

Why Hyperparameter Tuning?

This step ensures that each model is optimized to achieve the best possible performance, preventing underfitting and overfitting by finding ideal parameter settings.

Results and Interpretation

93.9

Accuracy

The model achieved an accuracy of 85%, indicating a high level of correctness in its predictions.

0.94

Weighted F1-Score

The model's weighted F1-Score is 0.82, reflecting a balanced performance across different classes.

Random Forest

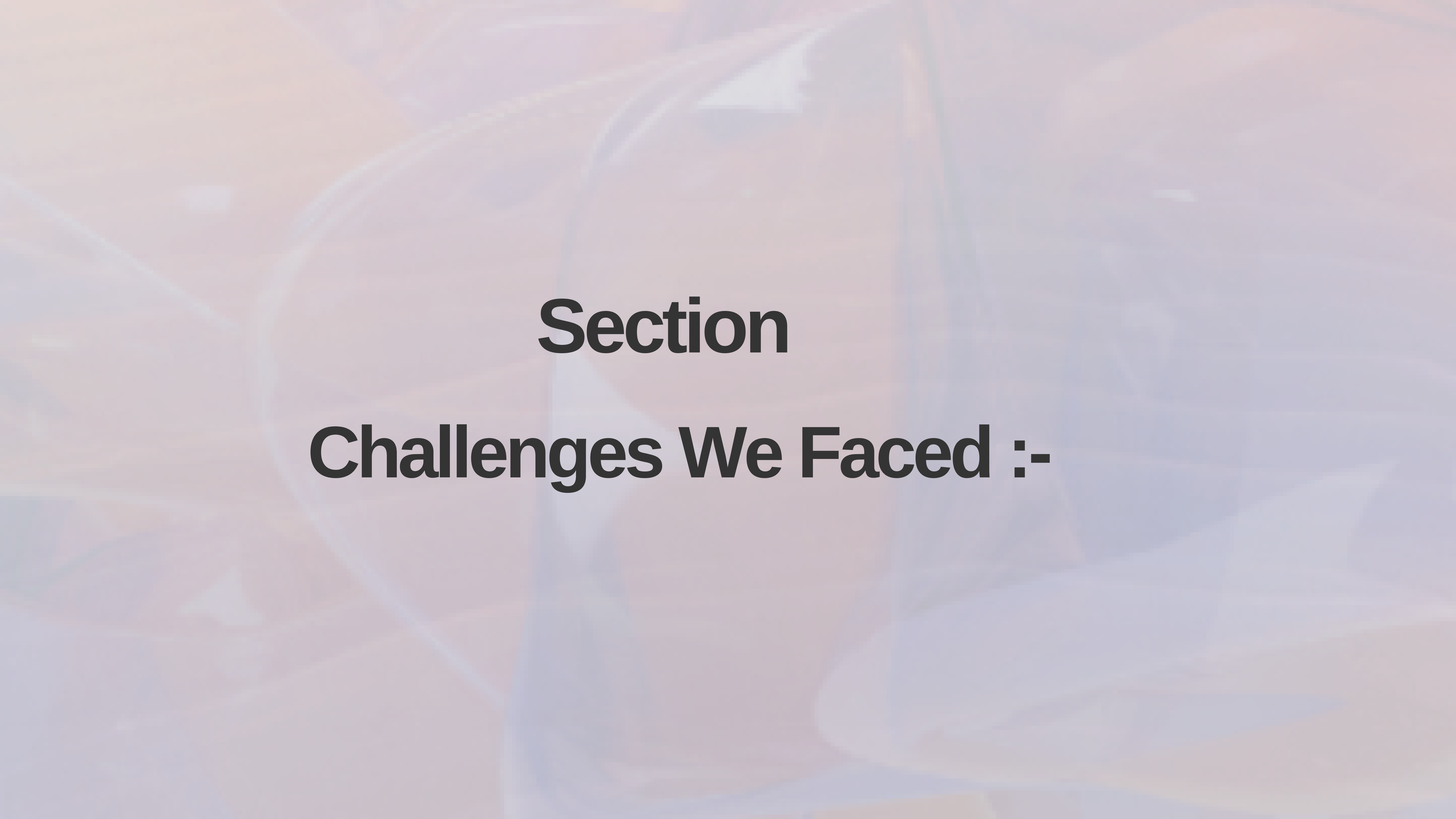
Best Model

The Random Forest model emerged as the top-performing model, showcasing high accuracy and F1-score due to its ensemble nature and ability to handle complex data structures.

Generalizability

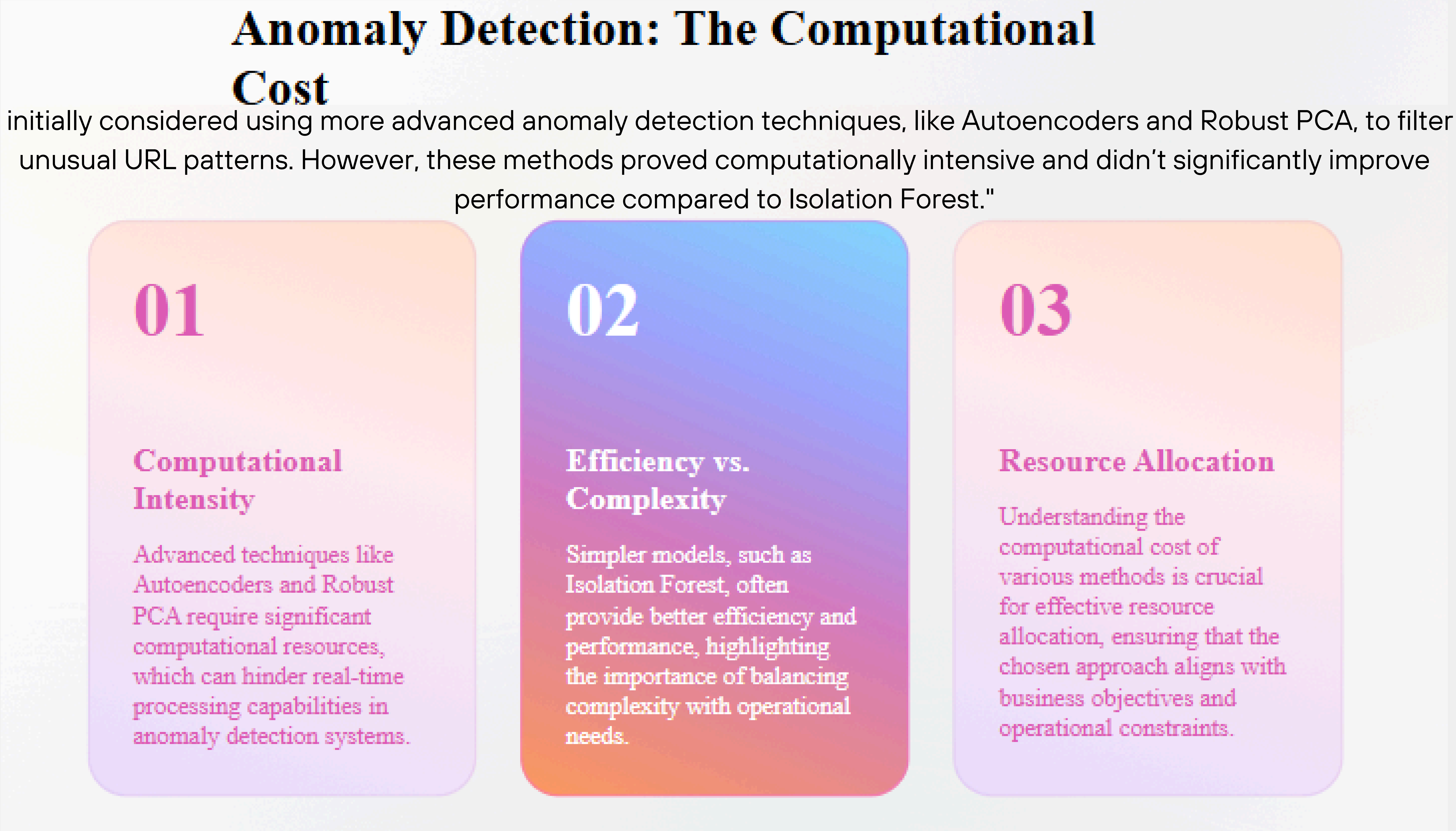
Result Interpretation

The Random Forest model demonstrated stable performance, indicating its ability to generalize well beyond this dataset.



Section

Challenges We Faced :-



Anomaly Detection: The Computational Cost

"We initially considered using more advanced anomaly detection techniques, like Autoencoders and Robust PCA, to filter out unusual URL patterns. However, these methods proved computationally intensive and didn’t significantly improve performance compared to Isolation Forest."

01

Computational Intensity

Advanced techniques like Autoencoders and Robust PCA require significant computational resources, which can hinder real-time processing capabilities in anomaly detection systems.

02

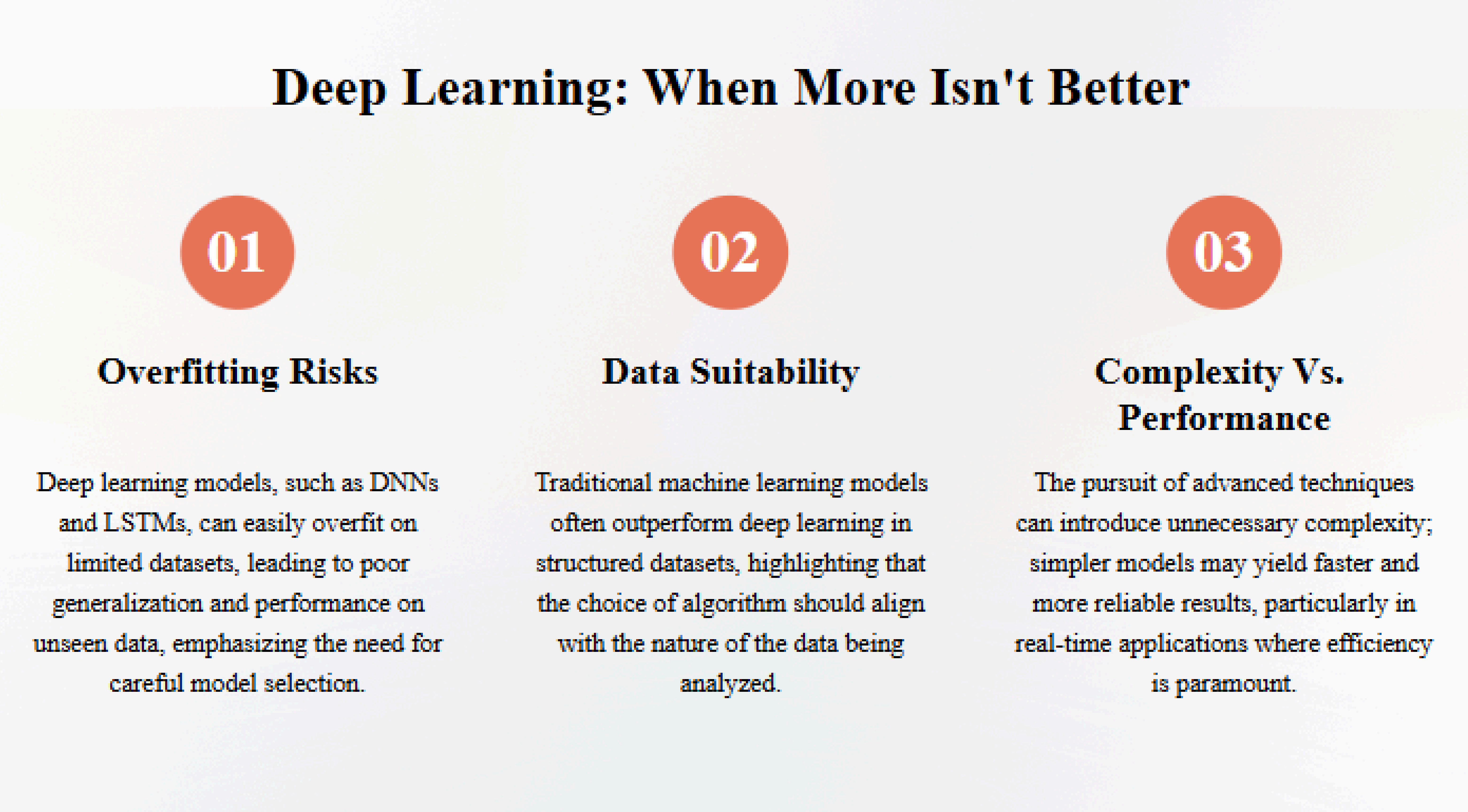
Efficiency vs. Complexity

Simpler models, such as Isolation Forest, often provide better efficiency and performance, highlighting the importance of balancing complexity with operational needs.

03

Resource Allocation

Understanding the computational cost of various methods is crucial for effective resource allocation, ensuring that the chosen approach aligns with business objectives and operational constraints.



Deep Learning: When More Isn't Better

01

Overfitting Risks

Deep learning models, such as DNNs and LSTMs, can easily overfit on limited datasets, leading to poor generalization and performance on unseen data, emphasizing the need for careful model selection.

02

Data Suitability

Traditional machine learning models often outperform deep learning in structured datasets, highlighting that the choice of algorithm should align with the nature of the data being analyzed.

03

Complexity Vs. Performance

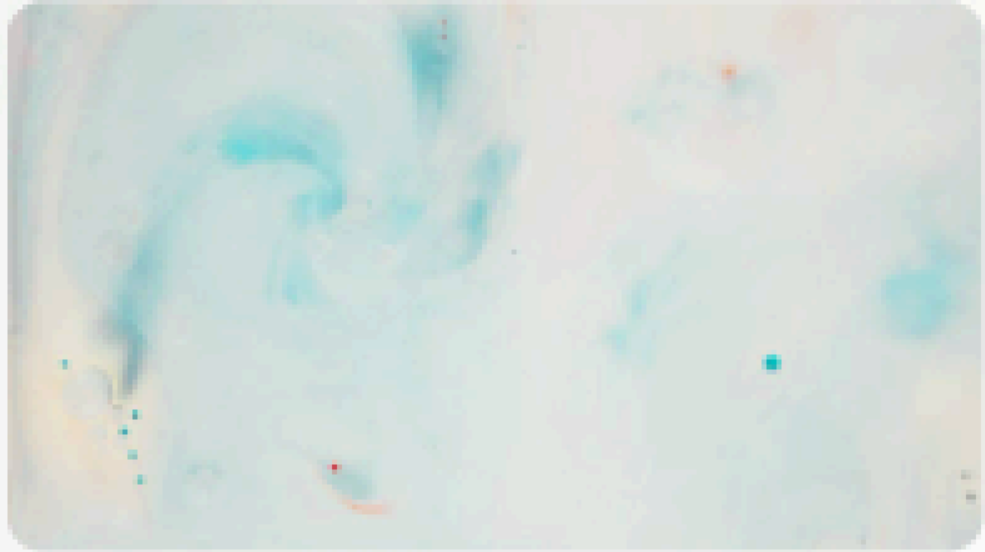
The pursuit of advanced techniques can introduce unnecessary complexity; simpler models may yield faster and more reliable results, particularly in real-time applications where efficiency is paramount.

We explored Deep Neural Networks and LSTM models to see if they could outperform traditional classifiers. But with limited data and a focus on structured features, these models didn't yield better accuracy and were prone to overfitting.

SMOTE: The Promise and the Reality

Synthetic Data Challenges

While SMOTE aims to balance class distribution by generating synthetic samples, it can inadvertently create data that does not accurately represent real-world scenarios, leading to potential model inaccuracies.



Tuning Requirements

Effective implementation of SMOTE requires meticulous tuning of parameters to ensure that the generated samples enhance model performance rather than introduce noise or bias into the dataset.

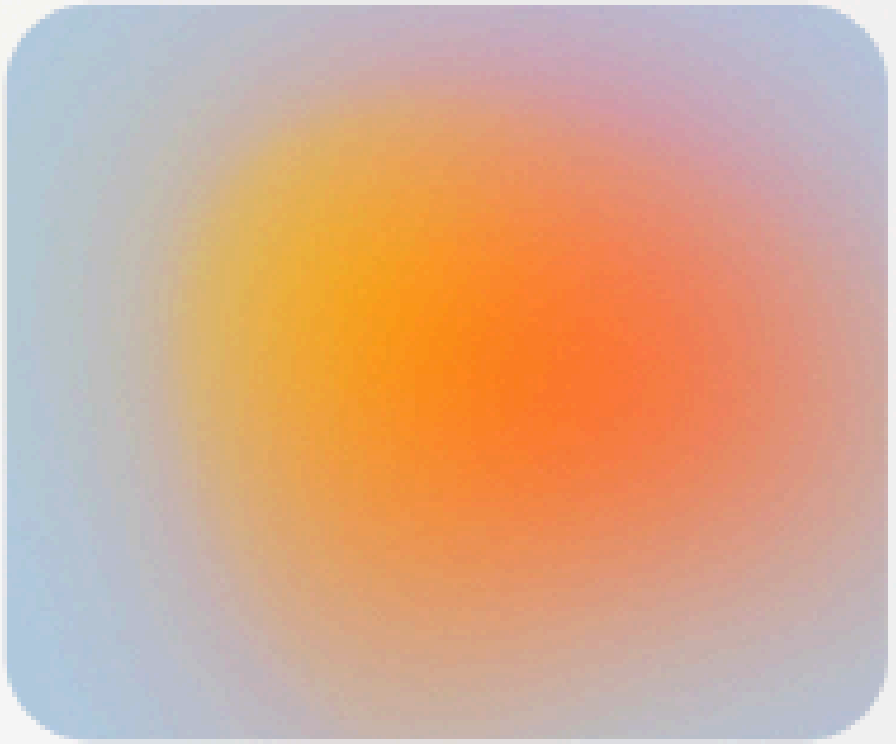
Alternative Approaches

Exploring alternative strategies, such as adjusting model weights or utilizing ensemble methods, may provide more reliable solutions for class imbalance without the pitfalls associated with synthetic data generation.

Lessons Learned: Simplifying Approaches

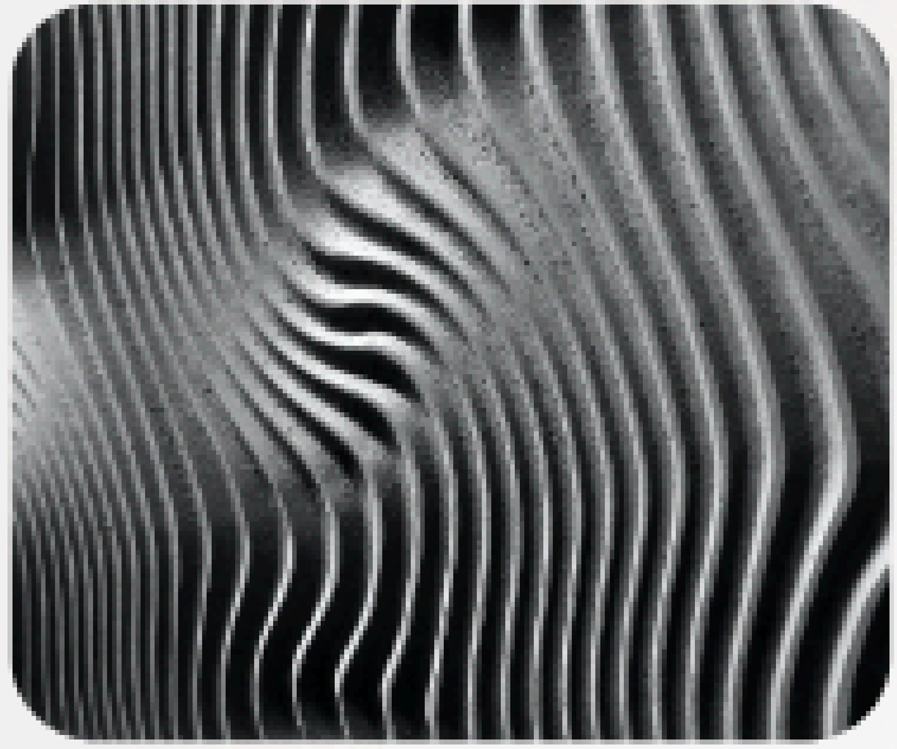
Efficiency Over Complexity

Emphasizing simpler methodologies can lead to improved efficiency, particularly in real-time applications where processing speed is crucial for timely decision-making and operational effectiveness.



Feature Quality is Key

Prioritizing the relevance and quality of features rather than their quantity can significantly enhance model performance, reducing noise and improving the clarity of insights derived from data.



Model Selection Matters

Choosing the right model based on data characteristics is essential; traditional machine learning approaches may outperform complex models in structured datasets, ensuring better accuracy and reliability.

Limitations and Future Work

01 Current Limitations

Anomaly Detection Limitations: Isolation Forest may not detect all complex patterns of obfuscation effectively.

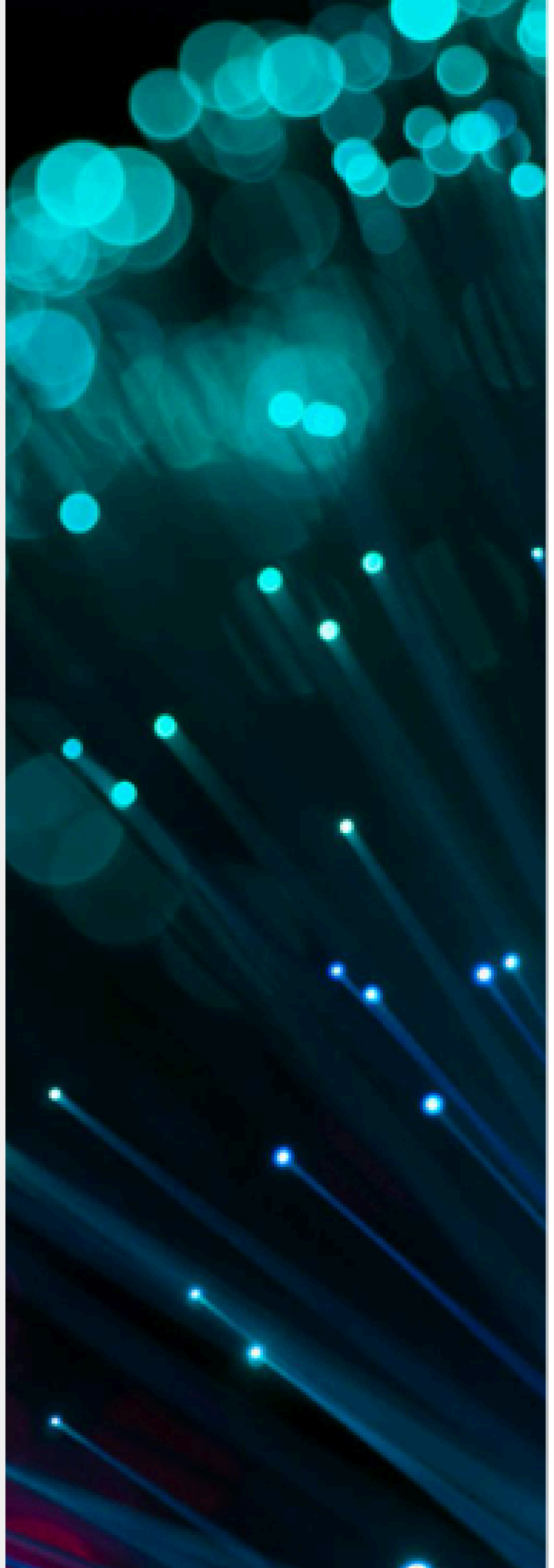
Class Imbalance: Less frequent URL types impact classifier robustness across all classes.

02 Future Enhancements

Advanced Anomaly Detection: Implement techniques like Robust PCA or Autoencoders to improve detection of sophisticated obfuscation patterns.

Additional Machine Learning Models: Explore complex classifiers like Support Vector Machines (SVM) and Gradient Boosting Machines (GBM) to capture different aspects of URL patterns.

Handling Class Imbalance: Utilize Synthetic Minority Over-sampling Technique (SMOTE) or cost-sensitive learning to improve accuracy for less frequent URL types.



Alternative Strategies for Class Balance

Exploring Ensemble Methods

Utilizing ensemble techniques, such as bagging and boosting, can enhance model robustness against class imbalance by combining multiple models to improve predictive performance on minority classes.

Cost-Sensitive Learning

Implementing cost-sensitive algorithms allows for the adjustment of misclassification costs, prioritizing the correct classification of underrepresented classes without relying solely on data augmentation techniques.

Adaptive Resampling Techniques

Employing adaptive resampling methods that dynamically adjust the sampling strategy based on model performance can lead to more effective handling of class imbalance, ensuring better representation of minority classes in training data.

Thank You