PDF Classifier using Trap4Phish Dataset

Shashank Mishra Shubham Negi Christopher Nyandoro

Table of Contents

Objective

Build an end-to-end ML pipeline which provided a PDF, can extract features from the PDF, give it as input to the ML Model, and then classify the PDF as malicious or benign, and also provide a confidence score, with a good recall score.

Data Collection: Trap4Phish PDF Dataset

- Contains 19296 rows of data
- Contains 42 features
- Benign PDF Source: Legitimate files collected from trusted public and institutional repositories (government data portals, academic datasets, open data sources).
- Malicious PDF Source: Files obtained from verified malware and phishing feeds such as MalwareBazaar, PhishTank, and VirusShare

Data Collection: Trap4Phish PDF Dataset

Features range from Structural Metadata, Content Objects, Behavioral Keywords, and Known Malicious Indicators

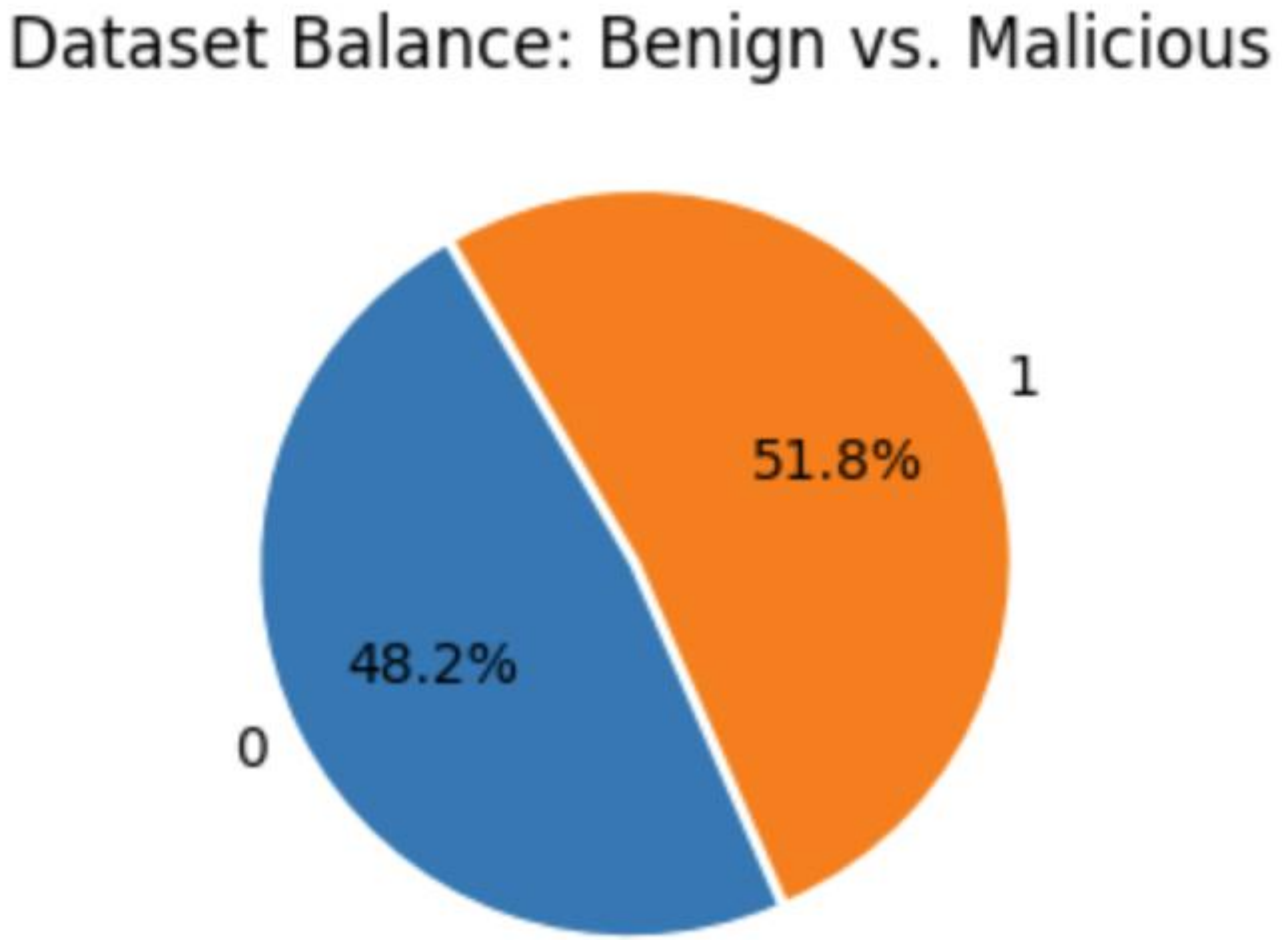
Examples

- Structural Metadata: file_size, page_count, title_chars, metadata_size
- Content Objects: object_count, stream_count, image_count
- Behavioral Keywords: javascript_count, openaction_count, aa_count
- Malicious Indicators: name_obfuscations, jbig2decode_count

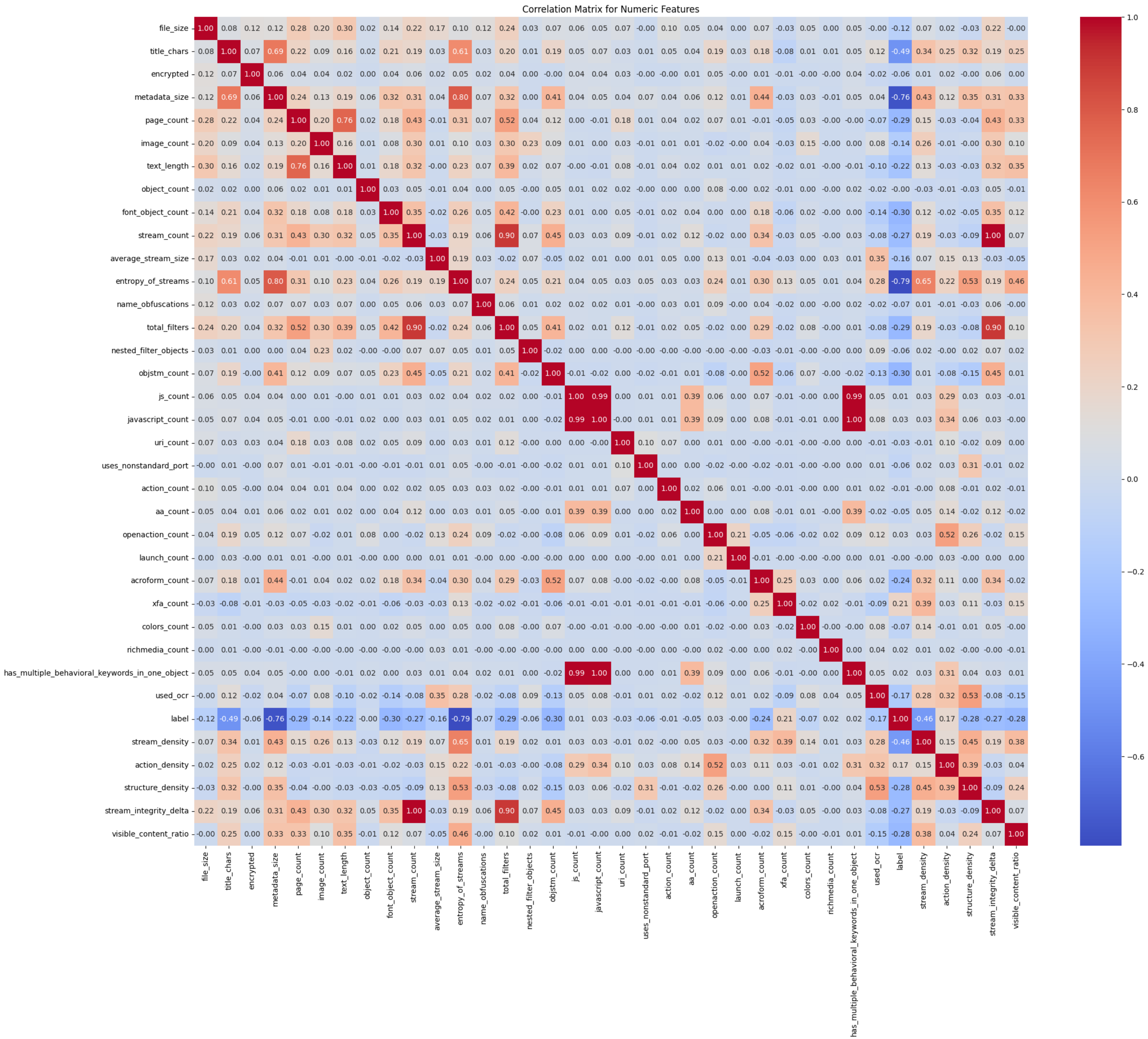
Exploratory Data Analysis (EDA)

- Class Distribution
- Feature Correlations
- ydata.profiling
- Box Plot for label
- Feature Importance using ML Classifier
- Scatter Plot
- Investigating the feature `used_ocr`

EDA: Class Distributions



EDA: Feature Correlations



EDA: ydata.profiling

Dataset statistics

Number of variables	37
Number of observations	19296
Missing cells	0
Missing cells (%)	0.0%
Total size in memory	5.4 MiB
Average record size in memory	296.0 B

Variable types

Numeric	37

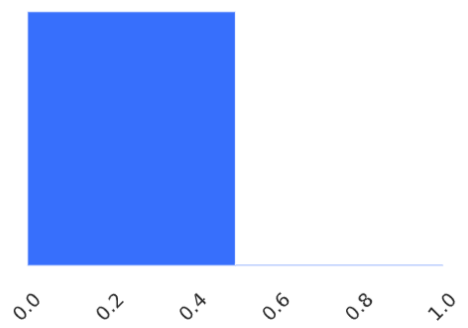
EDA: ydata.profiling – Problem: Rare Signals

encrypted

Real number (ℝ)

Zeros

Distinct	2	**Minimum**	0
Distinct (%)	< 0.1%	**Maximum**	1
Missing	0	**Zeros**	19212
Missing (%)	0.0%	**Zeros (%)**	99.6%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	0.004353233831	**Memory size**	150.9 KiB

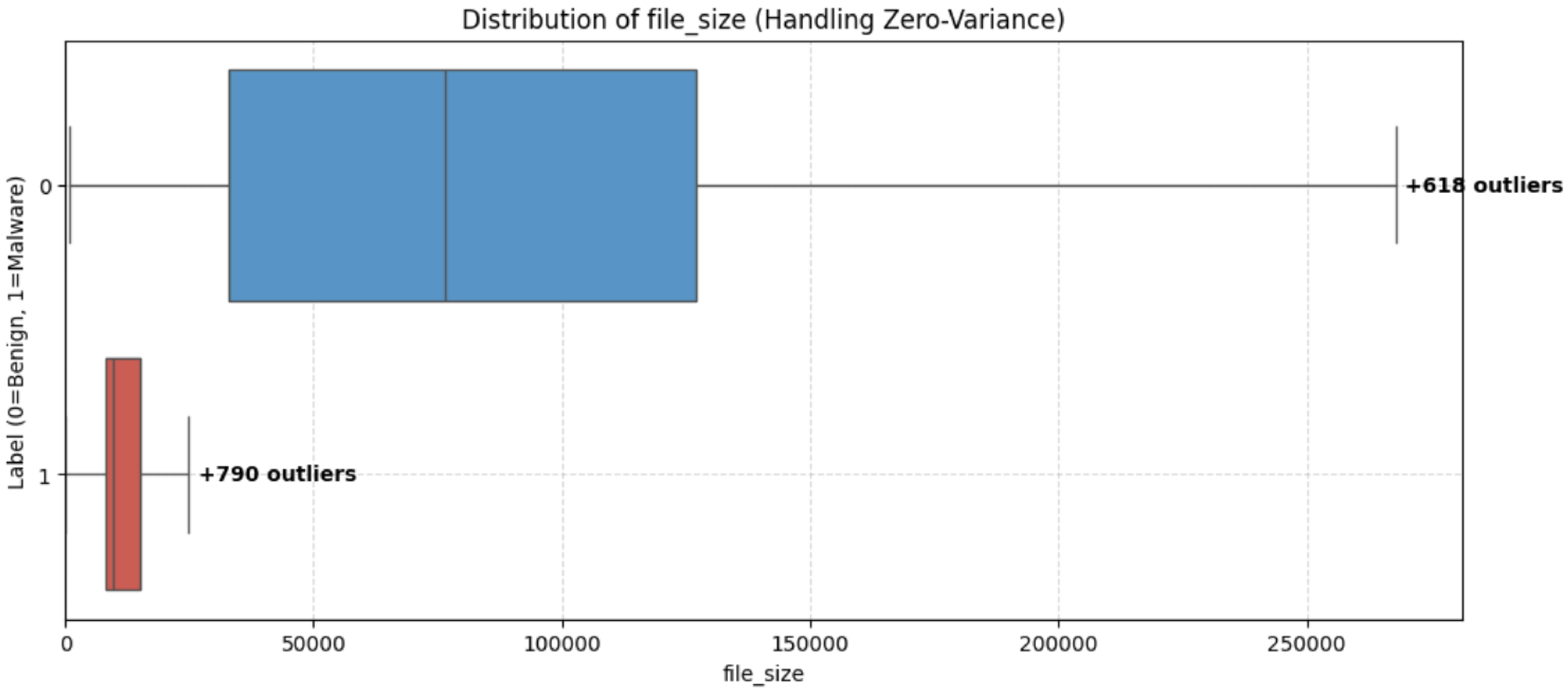


More details

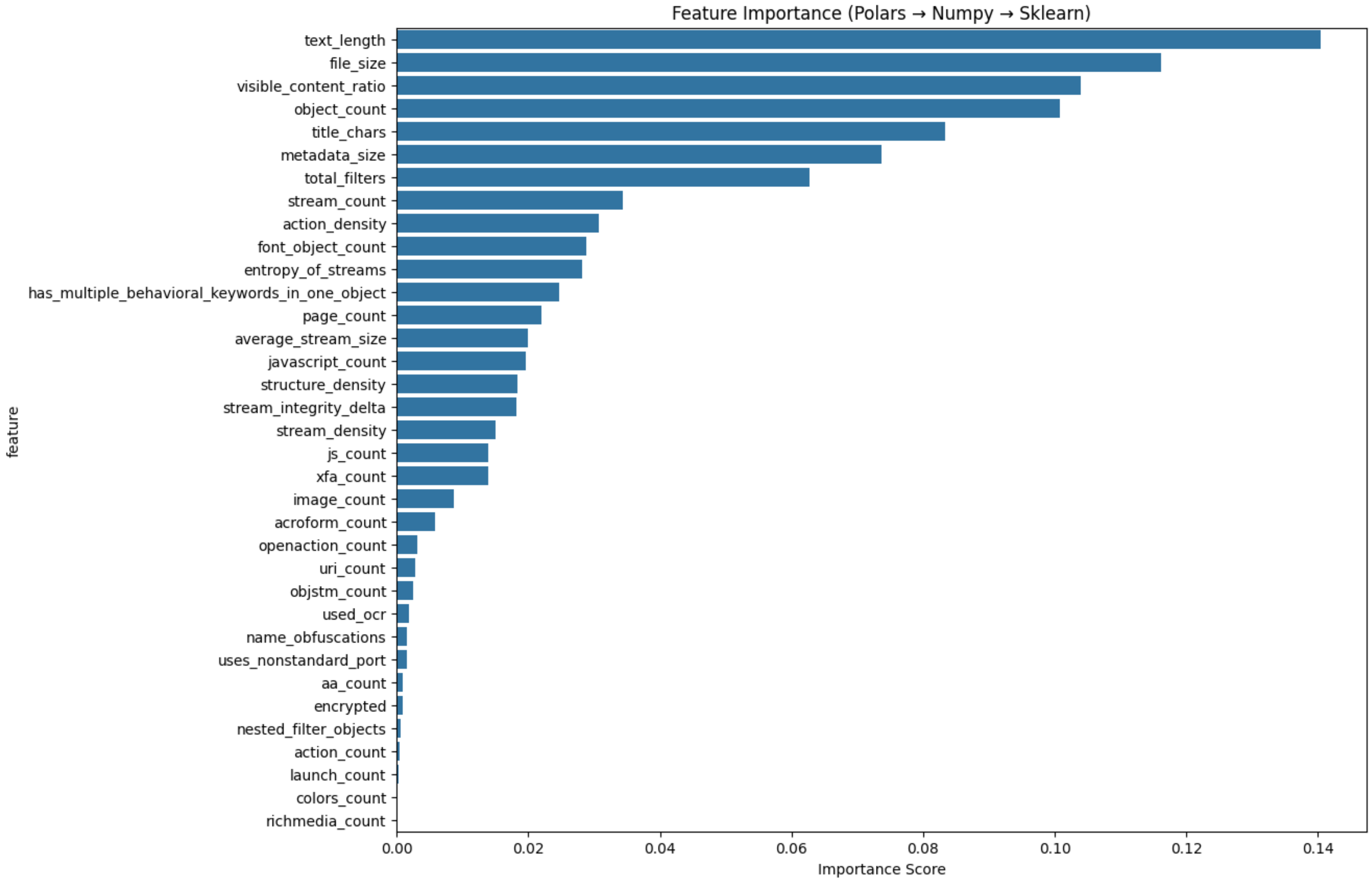
Statistics Histogram Common values Extreme values

Value	Count	Frequency (%)
0	19212	99.6%
1	84	0.4%

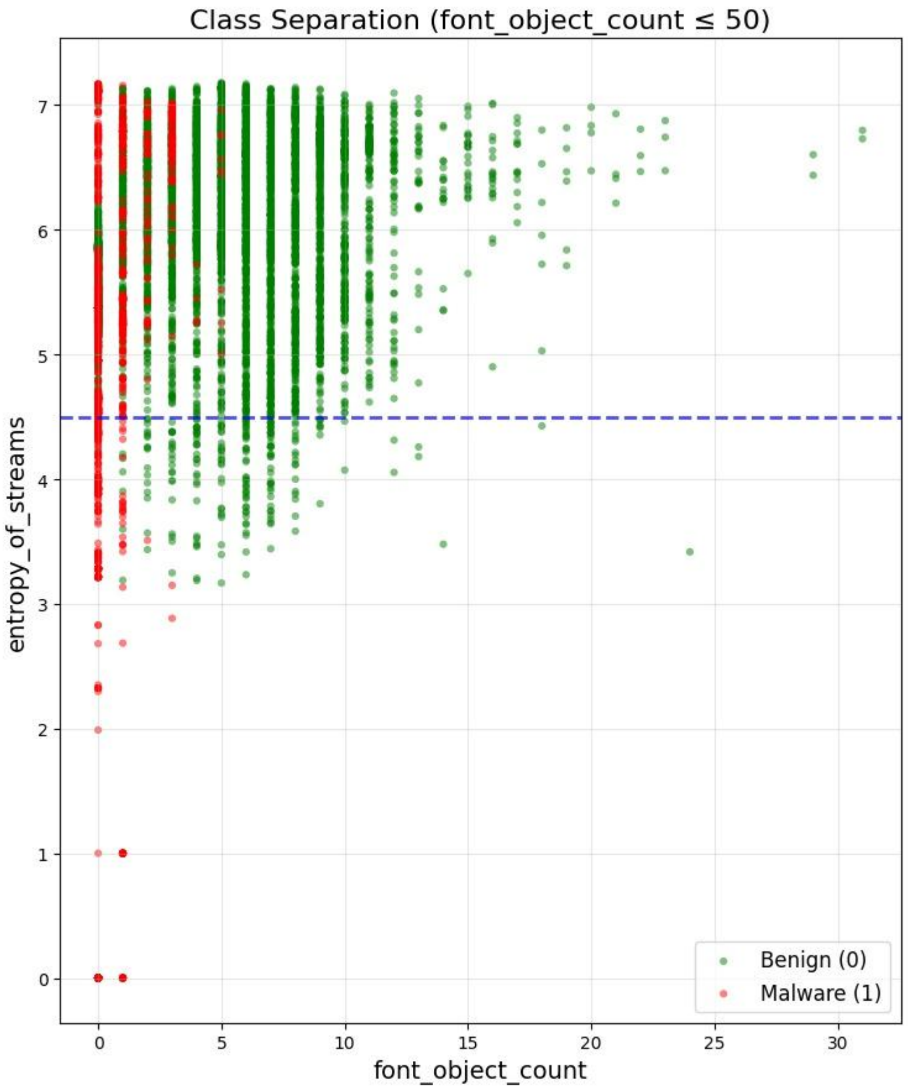
EDA: Box Plot


Distribution of file_size (Handling Zero-Variance)
+618 outliers
+790 outliers
Label (0=Benign, 1=Malware)
0
1
0
50000
100000
150000
200000
250000
file_size

EDA: Feature Importance using ML



EDA: Scatterplot – Clear Separation problem



entropy ≈ 4.5

EDA: Investigating the feature `used_ocr`

- Binary feature which tells if OCR used to extract features from the PDF.
- Cause of concern: Exception statement in Extraction Code

[Refer to ocr.ipynb]

Data Cleaning & Processing

- 1. Missing values -> Removed Empty Columns
- 2. Duplicates -> Duplicates removed
- 3. Inconsistences -> used_ocr investigated
- 4. Transformation -> log1p used to fix right-skeweness
- 5. Outlier Management -> Used IQR to Winsorize

[Refer to Model3.ipynb]

Cross Validation strategy

- **Strategy**: StratifiedKFold(n_splits=10, shuffle=True, random_state=42) preserves class ratios per fold.
- **Metrics tracked**: Recall (priority), Precision, F1, Accuracy, ROC-AUC.
- **Rationale**: Stratification mitigates imbalance bias; 5-fold balances variance and runtime.
- **Robustness checks**: Tried 10-fold and repeated stratified CV; results consistent.

Model Selection & Training

```
================================================================================
MODEL SELECTION & HYPERPARAMETER TUNING ANALYSIS
================================================================================

================================================================================
PART 1: COMPARING DIFFERENT ALGORITHMS
================================================================================

Evaluating models with 5-fold Stratified Cross-Validation...

--------------------------------------------------------------------------------
Model                       Mean Acc       Std   Time (s)
--------------------------------------------------------------------------------
Logistic Regression           0.9702    0.0044       7.88
Decision Tree                 0.9962    0.0012       0.34
Random Forest                 0.9975    0.0007       4.41
Gradient Boosting             0.9969    0.0007      10.44
K-Nearest Neighbors           0.9816    0.0016       0.99
Naive Bayes                   0.7023    0.0102       0.15
--------------------------------------------------------------------------------
```

- **Feature Standardization** (z-score normalization)
- **Stronger Regularization** (deeper pruning, higher min_samples)
- **Simpler Model** (fewer trees, shallower depth)
- **Lower decision threshold** for malware detection

```
5️⃣  KEY INSIGHTS
────────────────────────────────────────────────────────────
✓ Tree-based models (RF, GB) significantly outperform linear models
✓ Hyperparameter tuning provides marginal improvement (~0.1-0.3%)
✓ High accuracy is consistent across different CV strategies
✓ Model generalizes well (minimal overfitting)
✓ All evaluation metrics exceed 99% - excellent performance
```

Hyperparameters Tuned

- `n_estimators`: More trees improve stability; chosen in the 100–200 range.
- `max_depth`: Cap to prevent memorization; chosen around 10–15 based on CV.
- `min_samples_leaf`: Small but >1 (e.g., 3–5) to reduce variance and catch minority patterns.
- `min_samples_split`: 5–10 to avoid overly fine splits.
- `max_features`: `'sqrt'` for decorrelated trees, improving generalization.
- `class_weight`: `{0: 1, 1: 3}` to penalize malware misclassification; raises recall with acceptable precision loss.

Optuna Tuning

- **Objective**: Maximize CV recall on class 1 (malware).
- **Sampler/Pruner**: `TPE` sampler with `MedianPruner` for early stopping of weak trials.
- **Search space**: `n_estimators` (50–200), `max_depth` (5–20 or `None`), `min_samples_leaf` (1–10), `min_samples_split` (2–20), `max_features` (`'sqrt'`, `'log2'`, 0.5), `class_weight` (e.g., `{0:1,1:2..5}`).
- **Outcome**: Efficient exploration converging on recall-focused configs; similar/better recall than grid/random with fewer trials.

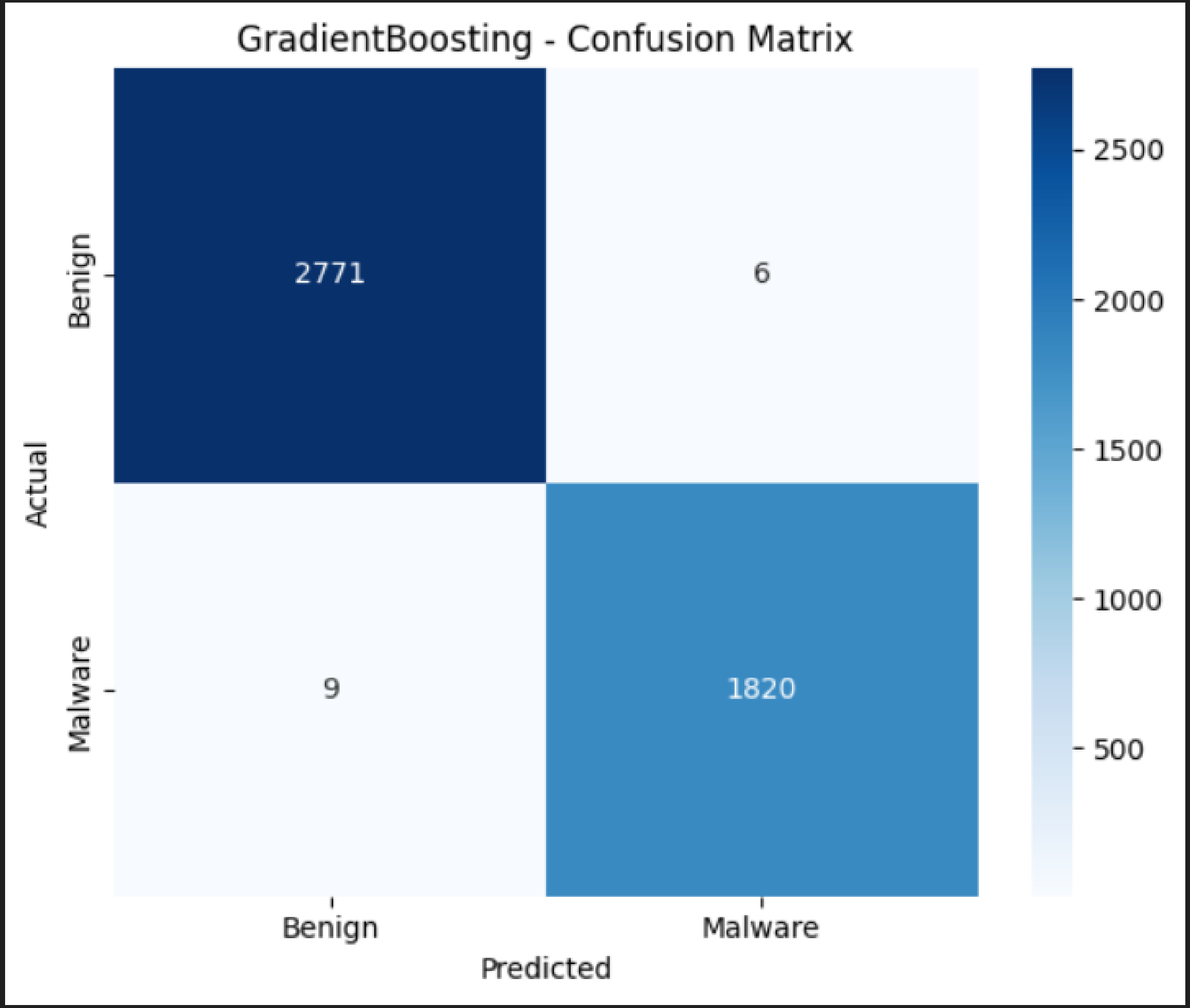
Class Weighting & Threshold

- **Class weighting**: Prefer `{0:1, 1:3}` as balanced point; higher weights (≥5) can overinflate FPs.
- **Threshold tuning**: Move decision threshold below 0.5 (e.g., ≈0.15) to reduce false negatives; select via precision-recall curve.
- **Process**: Sweep thresholds, report FN/FP/Recall/Precision; pick threshold minimizing FN with acceptable FP.

Evaluation

- **Primary metric: Recall**: Optimize for catching malware; minimize false negatives over raw accuracy.
- **Cost model**: False negative is high-risk (missed malware); false positive is lower-cost (extra review).
- **Trade-off**: Accept more false positives if it materially reduces false negatives.

```
ROC-AUC Score: 0.9991
```



✅ No Leakage Detected

Check	Result
Features with |r| > 0.95	None found
Single dominant feature (>50% importance)	No — top feature has 16%
Train-test gap	0.06% (minimal)

Ablation Study Results

Features Removed	Accuracy
0 (baseline)	99.74%
Top 5	99.64%
Top 10	99.65%
Top 15	99.09%
Top 20	80.82%

Production Pipeline Implementation

Model Import & Prediction System

Production Pipeline Architecture

- src/
- ├── features.py # Extract 41 features from PDF
- ├── preprocessing.py # Transform pipeline (Winsorize → log1p → scale)
- ├── import_model.py # Import trained models
- └── predict.py # Prediction CLI

- models/ # Saved artifacts
- ├── model.joblib # The trained model
- ├── caps.json # Transformation parameters
- ├── scaler.joblib # Fitted StandardScaler
- └── features.json # Feature names

Core Design

- Goal: Import any trained model and ensure consistent preprocessing
- Challenge: Training preprocessing must match prediction preprocessing exactly
- Solution: Save all transformation parameters alongside model

preprocessing.py - The Pipeline

```
class PDFPreprocessingPipeline:
    def fit(self, df):
        # Training mode: Learn from data
        self.transformer.fit(df)        # Learn 99th percentile caps
        self.scaler = StandardScaler()  # Fit scaler

    def transform(self, features):
        # Prediction mode: Apply saved transformations
        # 1. Winsorize + log1p (using saved caps)
        # 2. Select 28 features
        # 3. Scale (using saved scaler)
        return X_scaled  # numpy array → model
```

Key Point: Same pipeline code for training and prediction, just different mode (fit vs transform)

import_model.py - Model Import

```
python src/import_model.py model.joblib --test
```

1. Load the trained model

2. Fit pipeline on training dataset → learns caps & scaler

3. Test compatibility (feature count, predictions)

4. Save everything to models/:
   - model.joblib
   - caps.json (transformation parameters)
   - scaler.joblib (fitted scaler)
   - features.json (feature names)

```
Sample predictions:
INFO -   ✅ Sample 0: True=Benign, Predicted=Benign
INFO -   ✅ Sample 1: True=Benign, Predicted=Benign
INFO -   ✅ Sample 2: True=Benign, Predicted=Benign
INFO -   ✅ Sample 3: True=Benign, Predicted=Benign
INFO -   ✅ Sample 4: True=Malicious, Predicted=Malicious
INFO -   ✅ Sample 5: True=Benign, Predicted=Benign
INFO -   ✅ Sample 6: True=Malicious, Predicted=Malicious
INFO -   ✅ Sample 7: True=Benign, Predicted=Benign
INFO -   ✅ Sample 8: True=Malicious, Predicted=Malicious
INFO -   ✅ Sample 9: True=Malicious, Predicted=Malicious
INFO -
Sample accuracy: 100.0%
INFO - ================================================================================
INFO -
Saving model to models...
INFO -   ✅ Model saved: models/model.joblib
INFO -   ✅ Features saved: models/features.json
INFO -   ✅ Transformation caps saved: models/caps.json
INFO -   ✅ Scaler saved: models/scaler.joblib
INFO -   ✅ Metadata saved: models/metadata.json
INFO -
================================================================================
INFO - IMPORT COMPLETE!
INFO - ================================================================================
INFO - Model directory: models
INFO -
You can now use this model with:
INFO -   python src/predict.py document.pdf
INFO - ================================================================================
```

```
INFO -
================================================================================
INFO - MODEL INSPECTION
INFO - ================================================================================
INFO - Model type: Pipeline
INFO - Expected features: 28
INFO - ================================================================================
INFO -
================================================================================
INFO - COMPATIBILITY CHECK
INFO - ================================================================================
INFO - Fitting preprocessing pipeline on datasets/PDF_All_features.csv...
INFO - Fitting transformer on 32 features...
INFO -   ✅ Learned caps for 32 features
INFO -   ✅ Scaler fitted on 28 features
INFO -   ✅ Pipeline fitted and caps learned
INFO - Our preprocessing outputs: 28 features
INFO - Model expects: 28 features
INFO - ✅ Feature count matches!
INFO - ================================================================================
INFO -
================================================================================
INFO - MODEL TESTING
INFO - ================================================================================
INFO - Loading test data from datasets/PDF_All_features.csv...
INFO -   Loaded 19296 samples
INFO - Applying our preprocessing...
INFO - Making predictions...
INFO -   ✅ Predictions successful!
INFO -
```

predict.py - Production Interface

```
python src/predict.py document.pdf
```

```
# 1. Load everything
model = load("models/model.joblib")
pipeline = PDFPreprocessingPipeline(
caps=load("models/caps.json"),
scaler=load("models/scaler.joblib")
)
# 2. Process PDF
raw_features = extractor.extract(pdf_path)    # 41 features
X = pipeline.transform(raw_features)        # 28 scaled features
# 3. Predict
probabilities = model.predict_proba(X)[0]
prediction = model.predict(X)[0]
```

Complete Flow

- Trained Model (from Model.ipynb)
- ↓
- import_model.py
- ├─ Fit pipeline on training data
- └─ Save: model + caps + scaler
- ↓
- predict.py
- ├─ Load: model + caps + scaler
- ├─ Extract features from PDF
- ├─ Apply same preprocessing
- └─ Predict: BENIGN/MALICIOUS

Result: Plug-and-play system for any trained model

Usage

- `# One-time: Import trained model`
- `python src/import_model.py model.joblib --test`

- `# Use: Predict on PDFs`
- `python src/predict.py document.pdf`
- `python src/predict.py document.pdf --json  # API mode`

Testing on Unseen Data

- Benign and Malicious PDFs were generated to test the models
- Benign PDFs were made using Microsoft Word
- Malicious PDFs were made using the DidierStevensSuite(https://github.com/DidierStevens/DidierStevensSuite)

file_path --- str	label --- str	malware_probability --- f64	prediction --- i64
pdf_files/test_output.pdf	Malware	0.999969	1
pdf_files/Shubham_Negi - Resum…	Benign	0.000071	0

Testing on Unseen Data

```
~/repositories/Trap4Phishing-ML pipeline *2 > python src/predict.py test_output.pdf
INFO - Loading model...
INFO - Using 28-feature (sparse removal) preprocessing mode
INFO - ✅ Model loaded
INFO - Analyzing: test_output.pdf
INFO -   [1/3] Extracting features...
WARNING - PyPDF2 extraction failed for test_output.pdf: [Errno 22] Invalid argument
WARNING - PyMuPDF extraction failed for test_output.pdf: bad xref
INFO -   [2/3] Applying preprocessing...
INFO -   [3/3] Running model inference...
================================================================================
🚨PREDICTION: MALICIOUS
================================================================================
File: test_output.pdf
Confidence: 95.3%

Probabilities:
  Benign:     4.7%
  Malicious: 95.3%
================================================================================
~/repositories/Trap4Phishing-ML pipeline *2 ?1 >
```

Testing on Unseen Data

```
~/repositories/Trap4Phishing-ML pipeline *2 !5 ?2 > python src/predict.py LearningAgreementSE.CNyandoro.pdf
INFO - Loading model...
INFO - Using 28-feature (sparse removal) preprocessing mode
INFO - ✅ Model loaded
INFO - Analyzing: LearningAgreementSE.CNyandoro.pdf
INFO -   [1/3] Extracting features...
WARNING - PyMuPDF extraction failed for LearningAgreementSE.CNyandoro.pdf: bad xref
INFO -   [2/3] Applying preprocessing...
INFO -   [3/3] Running model inference...
================================================================================
✅ PREDICTION: BENIGN
================================================================================
File: LearningAgreementSE.CNyandoro.pdf
Confidence: 100.0%

Probabilities:
  Benign:    100.0%
  Malicious: 0.0%
================================================================================
~/repositories/Trap4Phishing-ML pipeline *2 !5 ?2 >
```

Gracías por tu atención y tiempo

Dataset Reference

Nejati, N. et al. "A Comprehensive Multi-Format Malicious Attachment Dataset for Email Threat Detection.“ Canadian Institute for Cybersecurity (CIC), University of New Brunswick, 2025.