AICS Assignment 4: Feature Engineering & Visualization for Malware Detection

Course: AI in Cybersecurity
Assignment: #4 of 16
Points: 15 (out of 100 total)
Due Date: August 12, 2024 at 11:59 PM ET
Submission: Jupyter Notebook + Written Report

🎯 Learning Objectives

By completing this assignment, you will:

1. **Master feature visualization** using seaborn and matplotlib for cybersecurity data
2. **Identify and engineer important features** for malware classification
3. **Analyze outliers** and understand their relationship to labeled malware data
4. **Connect feature engineering** to machine learning model performance
5. **Develop skills** in detecting unlabeled malware samples through feature analysis

📋 Assignment Overview

You will work with the malware dataset (`dataset_malwares.csv`) to perform comprehensive feature engineering and visualization analysis. This assignment builds directly on the malware classification techniques covered in Class 07 and prepares you for the capstone project.

Core Focus Areas:

- Advanced data visualization with seaborn
- Statistical feature analysis and selection
- Outlier detection and analysis
- Feature engineering for improved model performance
- Practical application to unlabeled sample detection

📊 Part 1: Advanced Feature Visualization (4 points)

1.1 Distribution Analysis (1.5 points)

Create **professional-quality visualizations** using seaborn to analyze feature distributions:

Requirements:

- **Distribution plots** comparing malware vs. benign samples for at least 5 key features
- **Box plots** showing quartile distributions and outliers
- **Violin plots** revealing distribution shapes and density patterns
- **Correlation heatmaps** for feature relationships

Specific Deliverables:

# Example structure - implement these visualizations:

# 1. Feature distribution comparison (malware vs benign)

# 2. Outlier visualization using box plots

# 3. Feature correlation matrix with annotations

# 4. Entropy analysis across different malware families

Quality Standards:

- Clear, readable titles and axis labels
- Appropriate color schemes for accessibility
- Professional formatting with legends
- Meaningful insights extracted from each visualization

1.2 Feature Relationship Analysis (1.5 points)

Explore **complex relationships** between features:

Requirements:

- **Scatter plots** with regression lines for key feature pairs
- **Pair plots** showing multiple feature interactions
- **Categorical analysis** of discrete features (Machine type, Subsystem, etc.)
- **Temporal patterns** if version/timestamp features are available

Analysis Questions to Address:

- Which features show the strongest correlations with malware classification?
- Are there non-linear relationships that simple correlation might miss?
- Do certain feature combinations create clear decision boundaries?

1.3 Advanced Visualization Techniques (1 point)

Implement **sophisticated visualization methods**:

Requirements:

- **Principal Component Analysis (PCA)** visualization in 2D/3D space
- **t-SNE or UMAP** for dimensionality reduction and cluster visualization
- **Feature importance plots** from multiple model types
- **Interactive plots** using plotly (optional, bonus points)

🔍 Part 2: Feature Identification and Engineering (5 points)

2.1 Statistical Feature Selection (2 points)

Apply **multiple feature selection techniques**:

Methods to Implement:

1. **Univariate Selection:** Chi-square, ANOVA F-test, mutual information
2. **Model-based Selection:** Random Forest, Gradient Boosting feature importance
3. **Recursive Feature Elimination (RFE)** with cross-validation
4. **Correlation-based filtering** to remove redundant features

Deliverables:

- **Comparison table** ranking features by different selection methods
- **Venn diagram** showing overlap between selection techniques
- **Justification** for final feature set selection
- **Validation** using domain knowledge from cybersecurity

2.2 Domain-Specific Feature Engineering (2 points)

Create **new features** based on cybersecurity domain knowledge:

Required New Features (minimum 5):

1. **Entropy ratios:** Compare section entropies to identify packed regions
2. **Size ratios:** Code-to-data, import-to-export, etc.
3. **Suspicious API indicators:** Flag dangerous function combinations
4. **Version consistency scores:** Detect fake or missing version information
5. **Memory layout anomalies:** Unusual section alignments or addresses

Engineering Process:

#

```
 Example feature engineering template:

def create_security_features(df):
    """

    Create domain-specific features for malware detection

    """
    # Entropy-based features
    df['entropy_variance'] = df['SectionsMaxEntropy'] -
df['SectionsMinEntropy']
    df['high_entropy_flag'] = (df['SectionsMeanEntropy'] > 6.5).astype(int)

    # Size-based ratios

    df['code_to_image_ratio'] = df['SizeOfCode'] / (df['SizeOfImage'] + 1)

    # Add your additional features here...

    return df
```

2.3 Feature Impact Analysis (1 point)

Quantify how your engineered features improve model performance:

Analysis Requirements:

- **Before/after comparison** of model accuracy with original vs. engineered features
- **Feature ablation study:** Remove features one by one to measure impact
- **Statistical significance testing** of performance improvements
- **Computational cost analysis** of new features

📈 Part 3: Outlier Analysis and Unlabeled Detection (3 points)

3.1 Outlier Detection Methods (1.5 points)

Implement **multiple outlier detection techniques**:

Methods to Apply:

1. **Statistical methods:** Z-score, IQR, modified Z-score
2. **Distance-based:** Local Outlier Factor (LOF), k-nearest neighbors
3. **Isolation-based:** Isolation Forest, One-Class SVM
4. **Ensemble methods:** Combine multiple outlier detection approaches

Analysis Requirements:

- **Compare outlier detection** across different methods
- **Relationship analysis:** How do outliers correlate with malware labels?
- **False positive analysis:** Which benign files are flagged as outliers?
- **Novel malware detection:** Can outliers indicate new malware families?

3.2 Unlabeled Sample Classification (1.5 points)

Apply your trained models to **detect potential malware** in unlabeled data:

Process Requirements:

1. **Create simulated unlabeled dataset** by removing labels from a subset
2. **Apply trained models** to predict malware probability
3. **Confidence analysis:** Identify high-confidence vs. uncertain predictions
4. **Outlier correlation:** Do outlier detection methods agree with ML predictions?

Evaluation Metrics:

- **Prediction confidence distributions**
- **Agreement between different detection methods**
- **Analysis of disagreement cases** - what makes them difficult to classify?

🤖 Part 4: Model Training Integration (3 points)

4.1 Feature Engineering Impact on Model Performance (1.5 points)

Systematically evaluate how feature engineering affects different ML algorithms:

Models to Compare:

- Logistic Regression (baseline)
- Random Forest
- Gradient Boosting
- Support Vector Machine

Analysis Framework:

```
# Performance comparison template:

def evaluate_feature_impact(original_features, engineered_features):
    """
    Compare model performance with different feature sets
    """
    results = {}
    for model_name, model in models.items():

        # Train with original features
        original_score = train_and_evaluate(model, original_features)

        # Train with engineered features
        engineered_score = train_and_evaluate(model, engineered_features)

        results[model_name] = {
            'original': original_score,
            'engineered': engineered_score,
            'improvement': engineered_score - original_score
        }
return results
```

4.2 Feature Selection Optimization (1.5 points)

Optimize your feature set for maximum model performance:

Requirements:

- **Grid search** over different numbers of features (10, 20, 30, 40, 50)
- **Cross-validation** to ensure robust performance estimates
- **Learning curves** showing performance vs. training set size
- **Feature stability analysis** across different data splits

Deliverables:

- **Optimal feature count** determination with justification
- **Final feature set** with cybersecurity interpretation
- **Performance comparison** across all tested configurations

📝 Written Report Requirements

Submit a **comprehensive written report in Markdown in your notebook** addressing:

Section 1: Executive Summary (0.5 pages)

- **Key findings** from your analysis
- **Best performing feature set** and model combination
- **Practical recommendations** for malware detection systems

Section 2: Visualization Analysis (1.5 pages)

- **Interpretation** of your key visualizations
- **Pattern identification** in malware vs. benign feature distributions
- **Insights** that would be valuable to cybersecurity analysts

Section 3: Feature Engineering Analysis (1.5 pages)

- **Justification** for your engineered features based on domain knowledge
- **Performance impact** quantification and statistical significance
- **Cybersecurity relevance** of your most important features

Section 4: Outlier and Detection Analysis (1 page)

- **Outlier patterns** and their relationship to malware classification
- **Unlabeled detection** methodology and results
- **Real-world applicability** for detecting unknown threats

Section 5: Model Performance and Recommendations (1 page)

- **Comprehensive performance comparison** across models and feature sets
- **Production deployment recommendations** with justification
- **Future improvements** and advanced techniques to explore

Section 6: Critical Analysis and Limitations (0.5 pages)

- **Limitations** of your approach and dataset
- **Potential biases** and how they might affect real-world performance
- **Adversarial considerations** - how might attackers evade your models?

🚀 Bonus Opportunities (up to 2 extra points)

Advanced Techniques (1 point each):

- **Deep feature learning:** Use autoencoders for feature extraction
- **Adversarial analysis:** Test model robustness against evasion attacks
- **Time series analysis:** If temporal features available, analyze malware evolution
- **Interactive dashboard:** Create a web-based tool for malware analysis

Code Requirements:

- **Clean, commented code** with clear variable names
- **Reproducible results** with random seeds set
- **Error handling** for common data issues
- **Performance timing** for computationally expensive operations

Visualization Requirements:

- **Professional quality** plots suitable for presentation
- **High resolution** (300 DPI minimum) for report inclusion
- **Consistent styling** across all visualizations
- **Colorblind-friendly** color schemes

📊 Grading Rubric (15 points total)

Component	**Excellent (A)**	**Good (B)**	**Satisfactory (C)**	**Needs Improvement (D/F)**
Visualization Quality (4 pts)	Professional plots with clear insights, excellent use of seaborn	Good visualizations with minor issues	Basic plots that convey information	Poor quality or uninformative plots
Feature Engineering (5 pts)	Creative, domain-informed features with clear performance impact	Solid feature engineering with some impact	Basic feature creation with minimal analysis	Limited or ineffective feature engineering
Outlier Analysis (3 pts)	Comprehensive outlier analysis with multiple methods and insights	Good outlier detection with some interpretation	Basic outlier identification	Superficial or incorrect outlier analysis
Model Integration (3 pts)	Systematic evaluation of feature impact across multiple models	Good model comparison with feature analysis	Basic model training with limited comparison	Poor model evaluation or missing analysis

Quality Indicators:

A-Level Work:

- Demonstrates deep understanding of cybersecurity domain
- Creates novel, effective features based on malware analysis principles
- Provides actionable insights for real-world deployment
- Shows creative problem-solving and critical thinking

B-Level Work:

- Solid technical execution with good domain understanding
- Effective feature engineering with measurable improvements
- Clear analysis and interpretation of results
- Professional presentation and documentation

C-Level Work:

- Meets basic requirements with adequate technical skills
- Some feature engineering with limited impact analysis
- Basic interpretation of results without deep insights
- Acceptable but not polished presentation

🛠️ Technical Setup and Resources

Required Libraries:

```python
# Data analysis and manipulation

import pandas as pd
import numpy as np

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px  # Optional for bonus

# Machine learning
from sklearn.ensemble import RandomForestClassifier, IsolationForest
from sklearn.svm import OneClassSVM
from sklearn.neighbors import LocalOutlierFactor
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.feature_selection import SelectKBest, RFE, mutual_info_classif
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.metrics import classification_report, confusion_matrix

# Statistical analysis
from scipy import stats
```

Dataset Information:

- **File:** `dataset_malwares.csv`
- **Size:** ~19,000 samples with 72 features
- **Target:** `Malware` column (0=benign, 1=malware)
- **Features:** PE file characteristics, entropy measures, size metrics

🆘 Support and Resources

Getting Help:

- **Office Hours:** Tuesdays and Thursdays, 2-4 PM ET
- **Discussion Forum:** Post questions for peer and instructor support
- **Email:** instructor@university.edu for urgent issues

Additional Resources:

- **PE File Format Documentation:** Microsoft PE/COFF specification
- **Malware Analysis Guides:** Practical Malware Analysis (book)
- **Seaborn Documentation:** https://seaborn.pydata.org/
- **Scikit-learn Feature Selection:** https://scikit-learn.org/stable/modules/feature_selection.html

Troubleshooting Guide:

- **Memory errors:** Reduce dataset size for testing, use chunking
- **Slow performance:** Start with smaller feature sets, optimize code
- **Visualization issues:** Check data types, handle missing values
- **Model convergence:** Adjust hyperparameters, check feature scaling

📅 Recommended Timeline

Week 1 (Aug 5-11):

- **Days 1-2:** Data exploration and basic visualizations
- **Days 3-4:** Feature selection and correlation analysis
- **Days 5-7:** Feature engineering and impact analysis

Week 2 (Aug 12):

- **Days 1-2:** Outlier analysis and unlabeled detection
- **Days 3-4:** Model training and performance evaluation
- **Days 5-6:** Report writing and final polishing
- **Day 7:** Final review and submission

Daily Time Commitment:

- **2-3 hours per day** recommended
- **Total effort:** 15-20 hours (appropriate for 15-point assignment)

⚖️ Academic Integrity

Collaboration Policy:

- **Encouraged:** Discussing concepts and debugging with peers
- **Citations Required:** Any external resources, tutorials, or code snippets

Originality Requirements:

- **Feature engineering** must be your own creative work
- **Analysis and insights** must reflect your own understanding
- **Code implementation** should be primarily your own

AI Tool Usage:

- **Allowed:** AI assistants for debugging and concept clarification
- **Required:** Document any AI tool usage in your submission

Good luck with your analysis! This assignment will give you valuable hands-on experience with the feature engineering and visualization skills essential for cybersecurity machine learning applications.