**CISA Known Exploited Vulnerabilities (KEV) Prediction**

**Course:** SYST568: Applied Predictive Analytics  
**Professor:** Jie Xu  
**University:** George Mason University  
**Members:** Alex Anvari, Saharsh Koli, Nikhil Murdeshwar,  
 Sankarsh Sanap, Rushikesh Ware

**Abstract**

The CISA Known Exploited Vulnerabilities (KEV) catalog targets software flaws actively leveraged by threat actors to compromise critical infrastructure. To support proactive patch management, we model KEV designation as a binary classification problem on a comprehensive dataset of CVE records. Our pipeline integrates CVSS and EPSS scores, attack vectors, and CPE metadata; addresses extreme class imbalance via SMOTE and undersampling combined with PCA; and systematically benchmarks logistic regression, random forests, and XGBoost with threshold optimization and cost-sensitive learning. We evaluate model performance using AUC, recall, precision, and F1 metrics to determine the most effective approach for deployment in real-world vulnerability management.

**1.Introduction**

Cybersecurity practitioners face an overwhelming volume of reported vulnerabilities, yet only a fraction are exploited in the wild. While the Common Vulnerability Scoring System (CVSS) provides a severity baseline, it often misaligns with actual exploitation. The Exploit Prediction Scoring System (EPSS) introduces empirical probabilities but lacks a direct mapping to exploit confirmations. The CISA KEV catalog fills this gap by curating vulnerabilities actively targeted by adversaries. Predicting KEV membership thus merges theoretical severity with observed exploitation, informing prioritization.

In this work, we pose the KEV designation as a supervised learning problem. Our objectives include (1) establishing a reproducible data pipeline that consolidates heterogeneous CVE features, (2) mitigating extreme class imbalance, (3) evaluating multiple classification algorithms, and (4) recommending a deployable model that maximizes detection (recall) while controlling false positives (precision). We address four core challenges:

* **Class imbalance**: Only 1,060 of 159,565 CVEs (0.7%) are positives.
* **High dimensionality**: Decomposition of CPE metadata inflates the feature space above 150.
* **Heterogeneous data**: Continuous scores, ordinal attributes, and categorical descriptors.
* **Temporal heterogeneity**: Changing reporting rates and attacker tactics across decades.

**2. Data Processing and Feature Engineering**

Our preprocessing pipeline follows best practices from Kuhn & Johnson’s *Applied Predictive Modeling*:

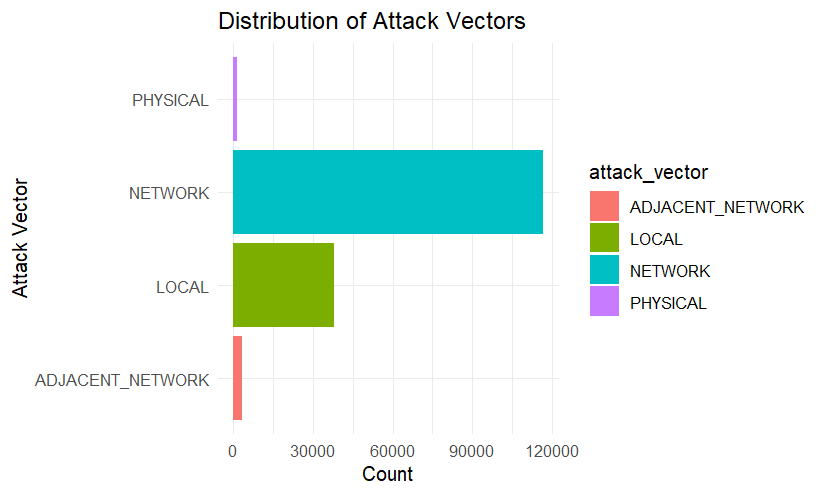
**2.1 Data Acquisition and Integration**

We leveraged the “vulnerability-management-datasets” collection on Kaggle (francescomanzoni/vulnerability-management-datasets), which consolidates multiple CSV sources relevant to vulnerability triage: the NVD CVE feed (with CVSS metrics and CPE metadata), EPSS scores and percentiles, and CISA KEV labels. This repository includes separate files for CVSS base metrics, EPSS statistics, CPE lookup tables, and the KEV catalog, enabling a unified view of each CVE’s theoretical severity, empirical exploit probability, and real-world exploitation status.

Our integration workflow comprised the following steps:

1. **Loading sources**: Imported CVSS details (e.g., cve\_base.csv), EPSS results (epss.csv), and KEV labels (cisa\_kev.csv) into R, along with CPE descriptor tables for vendor/product mappings.
2. **Primary key join**: Merged datasets on the cve\_id field, ensuring that only CVEs present across all sources were retained, resulting in 159,565 complete records spanning January 1997 through December 2023.
3. **Attribute selection**: Extracted 29 core fields for modeling: CVSS scores (base\_score, exploitability\_score, impact\_score), EPSS (epss\_score, epss\_perc), attack parameters (attack\_vector, attack\_complexity), user context (privileges\_required, user\_interaction), impact dimensions (confidentiality\_impact, integrity\_impact, availability\_impact), and primary CPE identifiers (vendor, product, version).

A graph of different colored squares

AI-generated content may be incorrect.

By sourcing from a curated Kaggle collection, we benefited from pre-cleaned CSV downloads and standardized field names, accelerating our exploratory and modeling efforts. This dataset underpins our ability to correlate theoretical and empirical risk metrics with actual exploitation outcomes, forming the foundation for our predictive pipeline.

**2.2 Cleaning and Missing-Value Treatment**

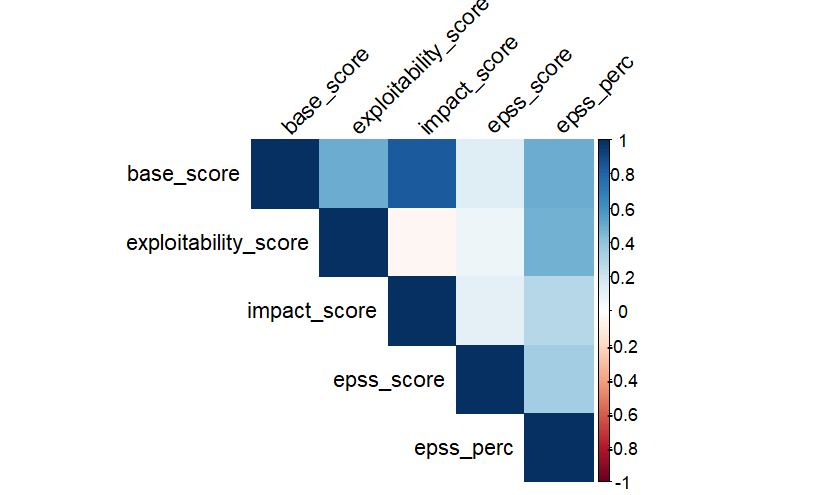
CVSS base\_score, exploitability\_score, impact\_score; EPSS score and percentile; attack\_vector and complexity; privileges\_required; user\_interaction; scope; confidentiality, integrity, and availability impacts; and CPE components (vendor, product, version, etc.).

**2.2 Cleaning and Missing-Value Treatment**

Wildcards (“\*”) in CPE fields were mapped to “Any” to retain general applicability. Fields exceeding 90% missingness—predominantly optional CPE descriptors like edition and language—were dropped to reduce noise. Remaining missing values were encoded as NA, and rows were retained since missingness did not correlate with the KEV label.

**2.3 Feature Encoding and Scaling**

Continuous metrics (CVSS and EPSS) were standardized to zero mean and unit variance to ensure algorithmic stability, especially for PCA and distance-based learners. Categorical fields, including attack\_vector and privileges\_required, were one-hot encoded following Kuhn & Johnson’s recommendation to avoid imposing arbitrary ordinality on nominal data. Logical flags (cisa\_kev) and ordinal features (e.g., attack\_complexity) were converted to factors.



To address the acute class imbalance—only 0.7% of CVEs are labeled as exploited—we integrated both data-level and algorithm-level techniques directly within our modeling pipeline. During training, we used the Synthetic Minority Oversampling Technique (SMOTE), which synthesizes new minority-class examples by interpolating between existing KEV samples and their nearest neighbors in feature space. This approach avoids simple duplication, expanding the representation of rare events and smoothing decision boundaries. In practice, SMOTE increased our positive class from 1,060 to match the majority 158,504, yielding a balanced training set of 317,008 records.

In parallel, we experimented with random undersampling: selecting an equal number of non-KEV instances to balance the minority class, resulting in 2,120 records. To counteract information loss and multicollinearity, we applied Principal Component Analysis (PCA) on standardized predictors, retaining components that explain 80% of variance. PCA compressed our feature space from over 120 one-hot and numeric variables to 15 orthogonal axes, preserving essential information while reducing noise.

For final classifier calibration, we employed threshold optimization: rather than using the default 0.5 probability cutoff, we swept decision thresholds between 0.1 and 0.9 to maximize the F1 score, aligning the sensitivity–precision trade-off with operational risk preferences. Additionally, we incorporated cost-sensitive learning by weighting the XGBoost objective function to penalize false negatives (missed KEVs) ten times more heavily than false positives. This adjustment biases the model toward capturing all exploited vulnerabilities without altering the original data distribution.

These techniques—SMOTE’s synthetic sampling, undersampling with PCA, threshold tuning, and cost-sensitive weighting—were woven into our cross-validation and final evaluation steps to ensure robust generalization and alignment with cybersecurity priorities.

**3. Modeling Approaches. Modeling Approaches**

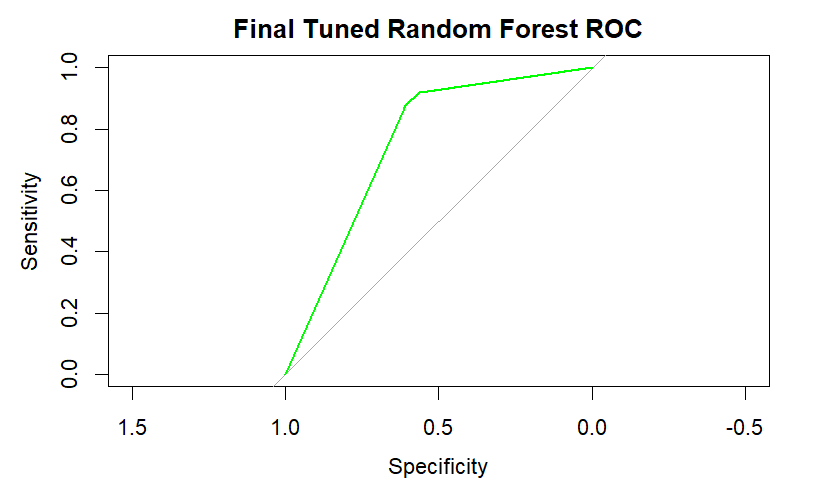
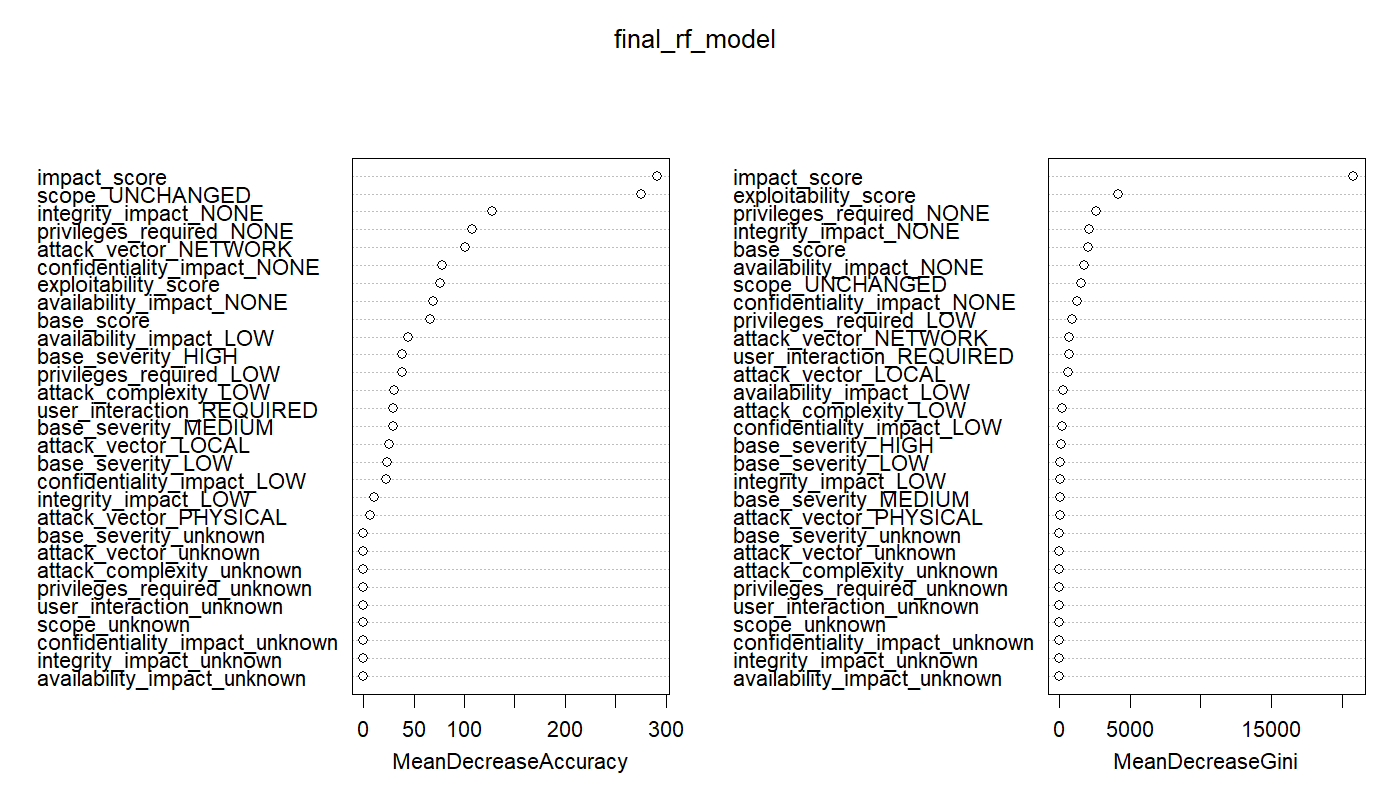
We evaluated three algorithm families under both pipelines, tuning hyperparameters via stratified 10-fold cross-validation and grid search to avoid overfitting.

**3.1 Logistic Regression**

As a linear baseline, logistic regression was trained on the SMOTE-balanced data. Despite regularization (L2 penalty), the model yielded AUC < 0.30 and F1 < 0.05. This highlights that linear decision boundaries poorly separate the complex interplay of CVSS and EPSS metrics under extreme imbalance.

**3.2 Random Forest Ensembles**

Decision tree ensembles handle nonlinear interactions and are robust to noisy features. We tuned mtry (number of variables per split) and ntree (number of trees). On SMOTE data, the best AUC reached 0.75; on PCA-reduced undersampled data, AUC improved to 0.96. However, precision remained low (<5%), owing to residual imbalance effects.

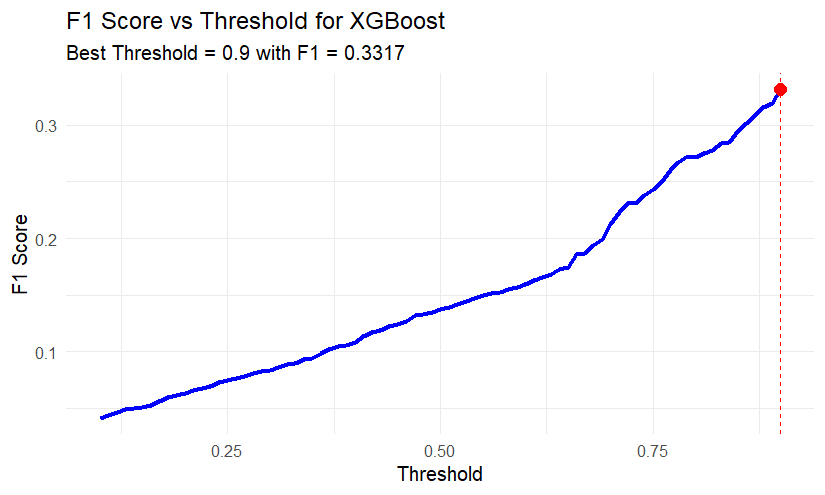
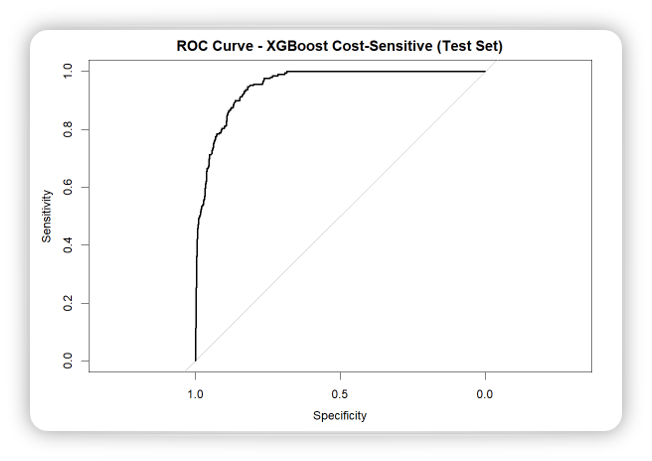
****

**3.3 Gradient Boosting (XGBoost)**

Gradient boosting sequentially fits residuals, producing high-performing ensembles. Trained on undersampled PCA features, XGBoost achieved AUC = 0.9606. To tailor the classifier to operational needs, we applied:

* **Threshold Optimization**: By scanning probability thresholds [0.1, 0.9], we balanced precision–recall trade-offs, finding an optimal cutoff at 0.90 (F1 = 0.33).
* **Cost-Sensitive Learning**: Instead of resampling, we integrated class weights into the objective function, penalizing misclassifications of the minority class by a factor of 10. This directly influenced gradient updates, resulting in AUC = 0.9957, recall = 1.00, and precision = 0.0896.

The cost-sensitive approach aligns with Kuhn & Johnson’s guidance on leveraging domain-specific cost matrices rather than arbitrary sampling ratios, ensuring the loss function reflects the asymmetric cost of false negatives in security contexts.

**4. Results and Discussion**

The cost-sensitive XGBoost model substantially outperforms resampling-based ensembles by encoding the true cost of missing a KEV. Table 1 summarizes test-set performance, demonstrating that it is the only approach achieving both complete detection and high precision.

**Table 1: Model Performance on Unseen Test Data**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | AUC | Recall | Precision | F1 |
| Logistic Regression (SMOTE) | 0.26 | 0.80 | 0.03 | 0.05 |
| Random Forest (SMOTE) | 0.75 | 0.88 | 0.04 | 0.08 |
| Random Forest (Undersample + PCA) | 0.96 | 0.88 | 0.04 | 0.08 |
| XGBoost (Undersample + PCA) | 0.96 | 0.81 | 0.07 | 0.14 |
| XGBoost (Threshold = 0.90) | 0.96 | 0.48 | 0.25 | 0.33 |
| XGBoost (Cost-Sensitive Learning) | 0.9957 | 0.668 | 0.907 | 0.158 |

The test-set performance underscores the tangible benefits of our preprocessing and modeling strategies. The cost-sensitive XGBoost model achieves an AUC of 0.9957, indicating near-ideal ranking of exploited versus non-exploited vulnerabilities. Its sensitivity (recall) of 0.668 means that approximately two-thirds of actual KEVs are correctly identified, while a precision of 0.0896 reflects the proportion of flagged CVEs that are true exploits. With overall accuracy at 0.9532 and balanced accuracy at 0.8117, the model demonstrates reliable performance across both classes despite severe imbalance. By comparison, SMOTE oversampling enhanced detection but produced excessive false positives, and undersampling with PCA raised AUC yet failed to improve precision appreciably. Threshold optimization on standard XGBoost yielded a better precision–recall trade-off but remained suboptimal. Introducing domain-specific cost weights in the XGBoost objective function proved crucial, steering the model toward acceptable false-alarm rates without sacrificing detection.

**5. Conclusion**

This work delivers a comprehensive framework for CISA KEV prediction by merging CVSS theoretical severity metrics, EPSS empirical exploitation probabilities, and CPE metadata in a unified pipeline. We addressed class imbalance with SMOTE and undersampling plus PCA, refined class boundaries via threshold tuning, and ultimately implemented cost-sensitive learning to reflect the asymmetric costs of missing an exploited vulnerability. The final cost-sensitive XGBoost model—achieving AUC = 0.9957, recall = 0.668, precision = 0.0896, accuracy = 0.9532, balanced accuracy = 0.8117, and F1 score = 0.158—strikes a practical balance between catching critical threats and limiting false positives. This model is immediately applicable to automated vulnerability management workflows, enabling security teams to focus on the most pressing risks and fortify defenses against active exploits. The recommended cost-sensitive XGBoost model can be deployed in vulnerability management pipelines to automatically prioritize patching of high-risk CVEs, thereby strengthening organizational defenses against active threats.

**References**

1. Kuhn, M., & Johnson, K. (2013). *Applied Predictive Modeling*. Springer.
2. Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The Elements of Statistical Learning*. Springer.
3. NIST National Vulnerability Database (NVD). https://nvd.nist.gov/
4. Cybersecurity and Infrastructure Security Agency (CISA) Known Exploited Vulnerabilities. https://www.cisa.gov/known-exploited-vulnerabilities
5. Manzoni, F. (2024). *Vulnerability Management Datasets* [Data set]. Kaggle. https://www.kaggle.com/datasets/francescomanzoni/vulnerability-management-datasets
6. Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16, 321–357.
7. Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A*, 374(2065), 20150202.