

BayesStrike – Notițe Matematice și Documentație (RO)

Proiect individual: Clasificator Bayes multinomial

Noiembrie 2025

1 Introducere

BayesStrike implementează un clasificator multinomial Naive Bayes pentru descrierile CVE (Common Vulnerabilities and Exposures). Scopul este etichetarea automată a descrierilor textuale în categoriile oficiale CVSS v3.1: **LOW**, **MEDIUM**, **HIGH**, **CRITICAL**. Documentul de față explică fiecare concept matematic pornind de la principii elementare: probabilități condiționate, presupunerea de independență „naive”, estimarea parametrilor pe baza datelor și modul în care aceste elemente se transformă în cod.

1.1 Terminologie de bază

- **Spațiul de etichete** \mathcal{Y} : mulțimea celor patru severități.
- **Vocabular** V : totalitatea token-urilor (cuvintelor) întâlnite după preprocesare.
- **Document** d : o descriere CVE, reprezentată ca vector de frecvențe \mathbf{c} .
- **Exemplu**: perechea (d, y) formată din descriere și eticheta ei reală.

2 Set de date și preprocesare

Prelucrarea datelor este esențială pentru ca ipotezele modelului să fie cât mai aproape de realitate.

1. **Încărcare**: `features.load_dataset` acceptă fișiere CSV (citite cu `DictReader`) și JSON (listă sau obiect cu cheie `records`). Fiecare element este normalizat într-o structură (`description`, `severity`).
2. **Maparea scorului** s : funcția `severity_from_score` aplică direct pragurile oficiale CVSS v3.1. Această mapare este justificată deoarece scorul numeric descrie același fenomen ca și eticheta discretă.
3. **Tokenizare**: definiția formală este $\text{tokenize}(t) = \text{regex_}/[a - z0 - 9] + /(t_{lower})$. Se elimină orice caracter nealfanumeric, lăsând doar secvențe consecutive.
4. **Stopword-uri**: mulțimea fixă S este folosită pentru a elimina termeni foarte frecvenți care nu contribuie la discriminare. Setul este stocat explicit în cod.
5. **Filtrarea documentelor goale**: dacă `preprocess_text` întoarce lista vidă, documentul este ignorat, iar contorul „skipped” este afișat utilizatorului.
6. **Împărțire stratificată**: `stratified_split` menține proporția fiecărei clase. Pentru fiecare bucket B_y se realizează `shuffle` determinist și se aleg primele $\lfloor 0.8 \cdot |B_y| \rfloor$ exemple pentru antrenare.

3 Modelul multinomial Naive Bayes

3.1 Probabilitate totală și Bayes

Pentru a calcula probabilitatea ca un document d să aparțină clasei y , plecăm de la formula lui Bayes:

$$P(y | d) = \frac{P(d | y)P(y)}{P(d)} = \frac{P(d | y)P(y)}{\sum_{y' \in \mathcal{Y}} P(d | y')P(y')}. \quad (1)$$

În practică, este suficientă comparația numerelor de la numărător, deoarece $P(d)$ este comun tuturor claselor.

3.2 Ipoteza „naivă”

Modelul presupune că toate token-urile w_i sunt condițional independente între ele odată ce clasa y este cunoscută. Astfel, dacă documentul conține c_i apariții ale token-ului w_i , distribuția multinomială ne dă

$$P(d | y) = \frac{(\sum_i c_i)!}{\prod_i c_i!} \prod_{i=1}^{|V|} P(w_i | y)^{c_i}. \quad (2)$$

Factorul combinatorial este identic pentru toate clasele, așa că îl putem ignora în comparație.

3.3 Reprezentare vectorială

Fie $\mathbf{c} = (c_1, \dots, c_{|V|})$ vectorul de frecvențe. Pentru eficiență, în implementare se folosește un `Counter` doar pe token-urile care apar efectiv, reducând complexitatea la numărul de cuvinte distincte din document.

3.4 Probabilități a priori

Numărul documentelor din clasă y este N_y , iar numărul total N . Prioriul este regularizat cu *add-one*:

$$P(y) = \frac{N_y + \alpha_0}{N + \alpha_0|\mathcal{Y}|}, \quad \alpha_0 = 1. \quad (3)$$

3.5 Probabilități condiționate și netezire

Numărul aparițiilor token-ului w_i în clasa y este $C_{y,i}$. Estimatorul maxim de verosimilitate ar fi $C_{y,i} / \sum_j C_{y,j}$, dar acesta devine zero pentru cuvinte neobservate. Pentru a evita log-probabilități $-\infty$, folosim netezirea Laplace (adăugăm un „pseudocount” α peste tot):

$$P(w_i | y) = \frac{C_{y,i} + \alpha}{\sum_j C_{y,j} + \alpha|V|}, \quad \alpha = 1. \quad (4)$$

Termenul $\alpha|V|$ din numitor poate fi interpretat ca o distribuție Dirichlet uniformă *a priori* asupra vocabularului.

3.6 Clasificare

Scorul logaritmic pentru un document este:

$$\log P(y | \mathbf{c}) \propto \log P(y) + \sum_{i=1}^{|V|} c_i \log P(w_i | y). \quad (5)$$

Predicția finală este $\hat{y} = \arg \max_y \log P(y | \mathbf{c})$. Pentru stabilitate numerică, codul lucrează integral în spațiul logaritmic.

4 Învățare și evaluare

4.1 Estimarea parametrilor

1. **Vocabularul:** `build_vocabulary_terms` parcurge fiecare document și înserează token-urile într-o mapare ordonată.
2. **Numărătoare de documente:** `class_doc_counts[y]` memorează N_y .
3. **Numărătoare de cuvinte:** matricea `class_word_counts[y][i]` furnizează $C_{y,i}$.
4. **Log-likelihood:** se precomputează $\log P(w_i | y)$ pentru a accelera predicțiile.

4.2 Măsuri de performanță

- **Acuratețe:** $\text{acc} = \frac{\sum_y M_{y,y}}{\sum_{i,j} M_{i,j}}$.
- **Precizie:** $\text{prec}_y = \frac{M_{y,y}}{\sum_i M_{i,y}}$.
- **Recall:** $\text{rec}_y = \frac{M_{y,y}}{\sum_j M_{y,j}}$.
- **F1:** $F1_y = 2 \cdot \frac{\text{prec}_y \cdot \text{rec}_y}{\text{prec}_y + \text{rec}_y}$.

Implementarea `evaluate_model` întoarce toate aceste valori și este folosită atât în CLI, cât și în dashboard-ul Flask.

4.3 Exemplu numeric

Presupunem că avem 100 de documente, dintre care 30 sunt HIGH. Atunci $P(\text{HIGH}) = (30 + 1)/(100+4) \approx 0.297$. Dacă token-ul „overflow” apare de 120 ori în clasa HIGH și suma frecvențelor tuturor token-urilor din HIGH este 10 000, atunci

$$P(\text{overflow} | \text{HIGH}) = \frac{120 + 1}{10\,000 + 1 \cdot |V|}.$$

Probabilitatea logaritmică este folosită pentru combinarea cu restul token-urilor.

5 Structura codului

- `features.py`: funcții principale (încărcare date, preprocesare, `train_pipeline`, `evaluate_existing_model`, `NaiveBayesModel`). Include subcomenzi CLI: `train`, `predict`, `evaluate`, `fetch-train`.
- `fetch_nvd.py`: client NVD 2.0, reutilizat direct de `features.py` `fetch-train`.
- `app/`: interfață Flask cu pagini Bootstrap pentru clasificare, antrenare și evaluare (folosește `run_web.py`).
- `tests/`: baterie `unittest` ce acoperă încărcarea datelor, preprocesarea, pipeline-ul de antrenare și salvarea rapoartelor.

6 Instrucțiuni de utilizare

1. **Pregătire date:** `python3 fetch_nvd.py --start-year 2022 --end-year 2022 --output data/cves.csv` sau pregătiți un CSV manual.
2. **Antrenare CLI:** `python3 features.py train --data data/cves.csv --model-path models/cve_nb`
3. **Predicție CLI:** `python3 features.py predict --model-path models/cve_nb.json "Buffer overflow"`
Comanda afișează eticheta și log-probabilitățile.
4. **Evaluare CLI:** `python3 features.py evaluate --data data/cves.csv --model-path models/cve_nb`
Fișierul JSON conține toate metricele.
5. **Fetch + train:** `python3 features.py fetch-train --start-year 2023 --end-year 2023 --output`
Acest pas automatizează întregul flux.

6. **Interfață web:** după ce există un model, rulați `python3 run_web.py` și lucrați din browser.

7 Referințe

- [R1] National Vulnerability Database API 2.0, <https://nvd.nist.gov/developers/vulnerabilities>.
- [R2] FIRST, *Common Vulnerability Scoring System v3.1 Specification Document*, 2019.
- [R3] T. M. Mitchell, *Machine Learning*, McGraw–Hill, 1997.