Apprentissage supervisé

Compte rendu projet Naives Bayes

Élèves :

Abir Jlassi

Jihene Guesmi

Omar Ben Rhouma

Enseignant :

Samir Toumi

Faculté de Sciences de Tunis

24 novembre 2024

Table des matières

1 Introduction

Ł'algorithme Naive Bayes est un modèle probabiliste simple mais puissant qui se situe dans le cadre de l'apprentissage supervisé. L'apprentissage supervisé est une méthode d'apprentissage automatique où l'algorithme est entraîné à partir d'un ensemble de données labellisées, c'est-à-dire un ensemble de données pour lequel chaque observation est associée à une étiquette ou une classe. L'objectif de l'apprentissage supervisé est de trouver une relation ou une règle permettant de prédire ces étiquettes pour de nouvelles données non étiquetées. Dans ce cadre, Naive Bayes est particulièrement adapté aux tâches de classification, où l'objectif est d'assigner à chaque observation une catégorie parmi un ensemble de classes prédéfinies. Basé sur le théorème de Bayes et une hypothèse d'indépendance conditionnelle entre les caractéristiques, Naive Bayes calcule la probabilité qu'une observation appartienne à chaque classe possible, et lui attribue la classe ayant la probabilité la plus élevée.

1.1 Exemples d'applications

Les applications de l'algorithme Naive Bayes sont nombreuses, en particulier dans les cas où des calculs rapides et une interprétation simple sont essentiels. Par exemple, cet algorithme est largement utilisé pour la détection de spam, l'analyse de sentiments, la classification de texte, la reconnaissance d'images, et la détection de fraudes. Il est souvent préféré pour ces tâches en raison de sa rapidité et de sa capacité à bien performer même avec des données de petite taille.

L'objectif de ce rapport est de :

1. **Présenter les concepts mathématiques de Naïve Bayes**, notamment le théorème de Bayes et l'hypothèse d'indépendance conditionnelle.
2. **Décrire ses variantes** pour s'adapter aux différents types de données.
3. **Illustrer son application pratique** avec des exemples théoriques et des implémentations en Python.
4. **Discuter ses forces et limitations** et ses cas d'utilisation concrets.

En comprenant les principes et les usages de Naïve Bayes, on espère montrer comment cet algorithme peut être un outil efficace et polyvalent pour diverses problématiques de classification dans l'apprentissage supervisé.

2 Concepts Mathématiques

L'algorithme Naive Bayes repose sur un principe fondamental en probabilité, connu sous le nom de théorème de Bayes.
Ce théorème permet de calculer la probabilité d'un événement A se produisant sachant qu'un autre événement B est déjà réalisé. En d'autres termes, il nous aide à déterminer la probabilité conditionnelle d'un événement en tenant compte de l'occurrence d'un autre événement.

2.1 Probabilité et Probabilité Conditionnelle

Avant d'énoncer le théorème, rappelons la différence essentielle entre **probabilité** et **probabilité conditionnelle**.

La probabilité simple (ou marginale) d'un événement A, notée P(A), indique la probabilité que cet événement se produise sans considérer d'autres informations.

En revanche, la **probabilité conditionnelle** de A étant donné B, notée P(AB), représente la probabilité de A sachant que B s'est produit. Cette notion de probabilité conditionnelle est cruciale dans les cas où nous cherchons à faire des prédictions en prenant en compte de nouvelles informations.

2.2 Enoncé du théorème de Bayes

Le théorème de Bayes nous fournit un moyen de relier ces deux types de probabilités, en exprimant la probabilité conditionnelle $P(A|B)$ en fonction de la probabilité inverse $P(B|A)$ et des probabilités marginales $P(A)$ et $P(B)$. Formellement, il s'énonce comme suit :

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{1}$$

Où :

- $P(A|B)$: **Posterior Probability** (probabilité a posteriori) est la probabilité de l'événement A sachant que B est réalisé,
- $P(B|A)$: **Likelihood Probability** (probabilité de vraisemblance) est la probabilité de l'événement B sachant que A est réalisé.
- $P(A)$: **Prior Probability** et P(B) "Marginal Probability" sont les probabilités marginales des événements A et B.

Dans le cadre de l'algorithme Naive Bayes, A représente généralement une classe que nous souhaitons prédire, et B représente un ensemble de caractéristiques ou d'observations sur lesquelles nous basons cette prédiction. Le théorème de Bayes devient alors un outil central pour calculer la probabilité qu'une observation appartienne à une classe donnée, en fonction des caractéristiques observées.

Dans les vraies applications du Naive Bayes, on calcule le résultat (Outcome) en se basant sur plusieurs variables. L'application du théorème de Bayes sur plusieurs variables rend le calcul complexe. Pour contourner cela, une approche consiste à prendre en considération ces variables indépendamment les unes des autres. Il s'agit d'une hypothèse forte. Généralement, les variables prédictives sont liées entre elles. Le terme « naïve » vient du fait qu'on suppose cette indépendance des variables.

2.3 Hypothèse d'Indépendance Conditionnelle

L'algorithme Naive Bayes suppose que chaque caractéristique est indépendante de toutes les autres, conditionnellement à la classe. En d'autres termes, pour des caractéristiques X1,X2,.,...,Xn, on fait l'hypothèse que :

$$P(X|C) = P(X_1|C) \cdot P(X_2|C) \cdot \ldots \cdot P(X_n|C)$$

Cette hypothèse simplifie énormément les calculs, même si elle est rarement vraie dans la pratique. Malgré cette hypothèse souvent irréaliste, le Naive Bayes fonctionne étonnamment bien dans de nombreux cas.

2.4 Formule du Classificateur Naive Bayes

À partir du théorème de Bayes et de l'hypothèse d'indépendance, on obtient le classificateur Naïve Bayes pour prédire la classe C d'une observation X comme suit :

$$C_{\text{prédit}} = \arg\max_{C} P(C) \prod_{i=1}^{n} P(X_i|C)$$

L'algorithme choisit la classe ayant la probabilité la plus élevée parmi toutes les classes possibles.

3 Variantes de l'algorithme Naive Bayes

Il existe plusieurs variantes de l'algorithme Naive Bayes, chacune adaptée à des types de données spécifiques :

- **Naive Bayes Gaussien :** Utilisé lorsque les caractéristiques sont continues. Il suppose que les caractéristiques suivent une distribution normale (ou gaussienne) pour chaque classe. La probabilité conditionnelle P(Xi / C) est alors calculée avec une fonction de densité gaussienne.
- **Naive Bayes Multinomial :** Couramment utilisé pour des données de type texte, il convient aux problèmes où les caractéristiques représentent le nombre de fois qu'un événement se produit (par exemple, le nombre de fois qu'un mot apparaît dans un document). Ce modèle est souvent appliqué à la classification de documents ou d'e-mails.
- **Naive Bayes Bernoulli :** Approprié pour des données binaires (par exemple, présence ou absence d'un mot dans un document). Ce modèle est particulièrement utile dans la classification de texte où chaque mot est traité comme une variable binaire.

Chacune de ces variantes adapte le calcul de P(Xi / C) aux caractéristiques des données, permettant ainsi d'obtenir des résultats plus précis en fonction des contextes d'application.

4 Exercice d'application mathématique

4.1 Classification de fruits (pomme, banane, orange)

Supposons que nous avons un ensemble de données d'entraînement avec des fruits étiquetés par leur type et leurs caractéristiques (couleur et taille).

— **Données d'entraînement :**

Fruit	**Couleur**	**Taille**
Pomme	Rouge	Grande
Banane	Jaune	Grande
Orange	Orange	Moyenne
Pomme	Vert	Moyenne
Orange	Orange	Moyenne
Banane	Jaune	Grande
Pomme	Rouge	Petite

TABLE 1 – Tableau des fruits, couleurs et tailles

— **Étapes de calcul pour classifier un fruit inconnu :**

Disons que nous avons un fruit inconnu qui est rouge et de taille moyenne, et nous voulons déterminer à quelle catégorie il appartient (pomme, banane ou orange).

Fruit X	**Rouge**	**Moyenne**

TABLE 2 – Caractéristiques des fruits

Étape 1 : Calcul des probabilités a priori :

*

Commençons par calculer la probabilité a priori de chaque catégorie, basée sur l'ensemble des données :

— $P(\text{Pomme}) = \frac{\text{nombre de pommes}}{\text{total des fruits}} = \frac{3}{7} \approx 0.43$

— $P(\text{Banane}) = \frac{\text{nombre de bananes}}{\text{total des fruits}} = \frac{2}{7} \approx 0.29$

— $P(\text{Orange}) = \frac{\text{nombre d'oranges}}{\text{total des fruits}} = \frac{2}{7} \approx 0.29$

*

Étape 2 : Calcul des probabilités conditionnelles :

Ensuite, pour chaque caractéristique (couleur et taille), calculons la probabilité conditionnelle pour chaque type de fruit.

Couleur	**Fruit**		
	Pomme	**Orange**	**Banane**
Rouge	$\frac{2}{3}$	0	0
Vert	$\frac{1}{3}$	0	0
Jaune	0	0	$\frac{2}{2}$
Orange	0	$\frac{2}{2}$	0

TABLE 3 – Probabilités conditionnelles pour la couleur des fruits

Probabilités de la couleur :

— $P(\text{Rouge}|\text{Pomme}) = \frac{\text{nombre de pommes rouges}}{\text{total des pommes}} = \frac{2}{3} \approx 0.67$

— $P(\text{Rouge}|\text{Banane}) = \frac{\text{nombre de bananes rouges}}{\text{total des bananes}} = 0$

— $P(\text{Rouge}|\text{Orange}) = \frac{\text{nombre d'oranges rouges}}{\text{total des oranges}} = 0$

Taille	Fruit		
	Pomme	**Orange**	**Banane**
Petite	$\frac{1}{3}$	0	0
Moyenne	$\frac{2}{3}$	$\frac{2}{2}$	0
Grande	0	0	$\frac{2}{2}$

TABLE 4 – Probabilités conditionnelles pour la couleur des fruits

Probabilités de la taille :

— $P(\text{Moyenne}|\text{Pomme}) = \frac{\text{nombre de pommes de taille moyenne}}{\text{total des pommes}} = \frac{1}{3} \approx 0.33$

— $P(\text{Moyenne}|\text{Banane}) = \frac{\text{nombre de bananes de taille moyenne}}{\text{total des bananes}} = 0$

— $P(\text{Moyenne}|\text{Orange}) = \frac{\text{nombre d'oranges de taille moyenne}}{\text{total des oranges}} = 1$

Étape 3 : Application du théorème de Bayes :

Calculons maintenant la probabilité de chaque type de fruit pour ce fruit inconnu.

Pour **Pomme** :

$$P(\text{Pomme}|\text{Rouge}, \text{Moyenne}) \propto P(\text{Rouge}|\text{Pomme}) \times P(\text{Moyenne}|\text{Pomme}) \times P(\text{Pomme})$$

En remplaçant les valeurs :

$$P(\text{Pomme}|\text{Rouge}, \text{Moyenne}) \propto 0.67 \times 0.33 \times 0.43 \approx 0.095$$

Pour **Banane :**

$$P(\text{Banane}|\text{Rouge}, \text{Moyenne}) \propto P(\text{Rouge}|\text{Banane}) \times P(\text{Moyenne}|\text{Banane}) \times P(\text{Banane}) = 0$$

Pour **Orange :**

$$P(\text{Orange}|\text{Rouge}, \text{Moyenne}) \propto P(\text{Rouge}|\text{Orange}) \times P(\text{Moyenne}|\text{Orange}) \times P(\text{Orange}) = 0$$

Étape 4 : Comparaison des probabilités :

Les probabilités pour chaque type de fruit sont les suivantes :

— $P(\text{Pomme}|\text{Rouge}, \text{Moyenne}) = 0.095$

— $P(\text{Banane}|\text{Rouge}, \text{Moyenne}) = 0$

— $P(\text{Orange}|\text{Rouge}, \text{Moyenne}) = 0$

La probabilité la plus élevée est celle de « pomme », donc on classifiera ce fruit inconnu comme une **pomme**.

4.2 Classification de genre en utilisant gaussian naive bayes(Male, female)

— **Données d'entraînement :**

Personne	Height (ft)	Weight (lbs)	Foot size (inches)
Male	6.00	180	12
Male	5.92	190	11
Male	5.58	170	12
Male	5.92	165	10
Female	5.00	100	6
Female	5.50	150	8
Female	5.42	130	7
Female	5.75	150	9

TABLE 5 – Tableau des caractéristiques des personnes

Étape 1 : Calcul des probabilités a priori

$$P(\text{Male}) = \frac{4}{8} = 0.5 \quad \text{et} \quad P(\text{Female}) = \frac{4}{8} = 0.5$$

Étape 2 : Calcul de l'espérance et de la variance

Male

$$\text{Espérance(Height)} = \frac{6 + 5.92 + 5.58 + 5.92}{4} = 5.855$$

$$\text{Variance(Height)} = \frac{(6 - 5.855)^2 + (5.92 - 5.855)^2 + (5.58 - 5.855)^2 + (5.92 - 5.855)^2}{4 - 1} = 0.035055$$

Genre	Espérance (Height)	Variance (Height)	Espérance (Weight)	Variance (Weight)	Espérance (Foot size)	Variance (Foot size)
Male	5.855	0.035033	176.25	122.92	11.25	0.91667
Female	5.4175	0.097225	132.5	558.33	7.5	1.6667

TABLE 6 – Tableau des espérances et variances pour Male et Female

Étape 3 : calcul pour classifier une personne inconnue Disons que nous avons une personne inconnue de taille 6 (ft), de poids 130 (lbs) et de pointure 8 (inch)

Genre	Height (ft)	Weight (lbs)	Foot size (inch)
Sample	6	130	8

TABLE 7 – Échantillon pour les calculs des probabilités

Posterior (Male) :

$$\text{Posterior (Male)} = P(M) \cdot P(H|M) \cdot P(W|M) \cdot P(FS|M)$$

$$P(\text{Male}) = \frac{4}{8} = 0.5$$

Posterior (Female) :

$$\text{Posterior (Female)} = P(F) \cdot P(H|F) \cdot P(W|F) \cdot P(FS|F)$$

$$P(\text{Female}) = \frac{4}{8} = 0.5$$

Densité de probabilité de la loi normale :

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$P(H|M) = \frac{1}{\sqrt{2 \times 3.142 \times 0.035033}} \times e^{-\frac{(6-5.855)^2}{2\times 0.035033}} = 1.5789$$

$$P(W|M) = 5.9881 \times 10^{-6}$$

$$P(FS|M) = 1.3112 \times 10^{-3}$$

$$P(H|F) = 2.2346 \times 10^{-1}$$

$$P(W|F) = 1.6789 \times 10^{-2}$$

$$P(FS|F) = 2.8669 \times 10^{-1}$$

Posterior (Male) :

$$P(M) \cdot P(H|M) \cdot P(W|M) \cdot P(FS|M) = 0.5 \times 1.5789 \times 5.9881 \times 10^{-6} = 6.1984 \times 10^{-9}$$

Posterior (Female) :

$$\mathbf{P(F)} \cdot \mathbf{P(H \mid F)} \cdot \mathbf{P(W \mid F)} \cdot \mathbf{P(FS \mid F)} = 0.5 \times 2.2346 \times 10^{-1} \times 1.6789 \times 10^{-2} \times 2.8669 \times 10^{-1} = 5.377 \times 10^{-4}$$

D'où l'individu sera une **female**.

5 Implémentation de l'algorithme en pratique

5.1 Introduction

Dans un monde de plus en plus connecté, la sécurité des utilisateurs en ligne est devenue une priorité majeure. L'un des risques les plus répandus est le phishing, une technique de cybercriminalité qui consiste à tromper les utilisateurs pour obtenir des informations sensibles, telles que des identifiants ou des données bancaires, en les redirigeant vers de fausses URL qui imitent des sites légitimes.

5.2 Cadre général du projet

Ce projet vise à développer un modèle capable de détecter automatiquement si une URL est légitime ou potentiellement liée à une attaque de phishing. Pour ce faire, nous exploitons le modèle de classification probabiliste Gaussian Naive Bayes, reconnu pour son efficacité dans les tâches de classification et sa simplicité d'implémentation.

Le projet s'inscrit dans une démarche de protection proactive des utilisateurs en ligne, avec des objectifs clairs :

- **Améliorer la détection des URL malveillantes :** Fournir un système capable d'identifier rapidement et efficacement les URL suspectes.
- **Réduire les faux positifs et faux négatifs :** Optimiser la précision de la classification pour garantir une utilisation fiable.
- **Renforcer les systèmes de sécurité existants :** Proposer une solution qui peut être intégrée dans des outils de sécurité comme les navigateurs web ou les filtres de messagerie.

Le choix du modèle Gaussian Naive Bayes repose sur sa capacité à gérer des données continues et à fournir des résultats rapides. À travers ce projet, nous espérons contribuer à la lutte contre le phishing en offrant une solution innovante et accessible.

5.3 Dataset et features

Pour développer un modèle capable de différencier les URL légitimes des URL de phishing, il est essentiel de disposer d'un dataset adéquat et de bien identifier les caractéristiques discriminantes. Cette section détaille les données et les caractéristiques utilisées pour le projet.

5.3.1 Dataset

Les URLs malveillantes représentent une menace sérieuse pour la cybersécurité. Ces sites peuvent héberger du contenu non sollicité tel que du spam, du phishing, des téléchargements à la volée (drive-by downloads), et d'autres attaques malicieuses. Ces techniques visent à tromper les utilisateurs, entraînant des pertes financières, le vol de données privées, ou encore l'installation de logiciels malveillants. Ce type d'activité engendre des pertes de plusieurs milliards de dollars chaque année.

Dans ce projet, nous utilisons un dataset volumineux et diversifié de 651 191 URLs, collecté à partir de plusieurs sources fiables et bien connues dans le domaine de la détection des URLs malveillantes. Le dataset est étiqueté en quatre catégories :

- **URLs bénignes (benign) :** 428 103 exemples d'URLs sûres.
- **URLs de defacement :** 96 457 exemples de sites altérés (défiguration).
- **URLs de phishing :** 94 111 exemples visant à tromper les utilisateurs pour voler leurs informations sensibles.
- **URLs de malware :**32 520 exemples contenant des logiciels malveillants.

L'objectif principal de cette dataset est de fournir une base robuste pour entraîner des modèles d'apprentissage automatique capables d'identifier les URLs malveillantes et de bloquer leur propagation avant qu'elles ne causent des dommages. La répartition des classes assure une bonne diversité des données, permettant une meilleure généralisation des modèles prédictifs.

5.3.2 Caractéristiques (Features)

Les caractéristiques extraites des URL jouent un rôle crucial dans la capacité du modèle à distinguer les URL légitimes des URL malveillantes. Voici les principales paramètres utilisées dans ce projet, ainsi que leur justification :

1. **Longueur de l'URL**

   Description : Les URLs de phishing ont tendance à être plus longues que les URLs légitimes pour cacher leur vraie nature.

   Caractéristique : Nombre de caractères dans l'URL.

2. **Nombre de points**

   Description : Les URLs de phishing contiennent souvent plusieurs sous-domaines ou séparateurs (points).

   Caractéristique : Nombre de points (`.`) dans l'URL.

3. **Utilisation d'une adresse IP**

   Description : Les URLs de phishing utilisent parfois une adresse IP au lieu d'un nom de domaine pour éviter la détection.

   Caractéristique : Vérification si le domaine est une adresse IP.

4. **Présence de caractères suspects**

   Description : Les symboles tels que `@`, `//`, `-`, et `=` dans l'URL peuvent indiquer une tentative de phishing.

   Caractéristique : Indicateur binaire pour chaque caractère suspect (par exemple, présence de `@`).

5. **Utilisation de HTTPS**

   Description : Les URLs de phishing manquent souvent de protocoles sécurisés comme HTTPS.

   Caractéristique : Vérification si l'URL utilise `https://`.

6. **Âge du domaine**

   Description : Les domaines de phishing sont généralement plus récents et de courte durée.

   Caractéristique : Âge du domaine (en mois ou années) basé sur les données WHOIS.

7. **Popularité du domaine**

   Description : Les domaines de phishing sont généralement impopulaires ou récemment enregistrés.

   Caractéristique : Vérification de la popularité à l'aide de sources comme le classement Alexa ou des services de réputation de domaine.

8. **Présence de redirections**

   Description : Les URLs de phishing peuvent utiliser des redirections (`//`) pour masquer leur destination réelle.

   Caractéristique : Nombre de motifs de redirection dans l'URL.

9. **Nombre de sous-domaines**

   Description : Les URLs de phishing contiennent souvent plusieurs sous-domaines pour paraître légitimes.

   Caractéristique : Comptage des sous-domaines (par exemple, `sub.sub.example.com` a 2 sous-domaines).

10. **Mots suspects dans l'URL**

    Description : Les mots tels que *"secure"*, *"login"*, *"account"*, *"update"*, etc., sont couramment utilisés dans les URLs de phishing.

    Caractéristique : Indicateur binaire pour chaque mot suspect.

11. **TLDs spéciaux (Top-Level Domains)**

    Description : Certaines URLs de phishing utilisent des TLDs moins courants.

    Caractéristique : Vérification si le TLD figure dans une liste de TLDs suspects (par exemple, `.xyz`, `.top`, etc.).

12. **Entropie de l'URL ou caractères inhabituels**

    Description : Les URLs de phishing peuvent utiliser des chaînes aléatoires ou des schémas à haute entropie pour éviter la détection.

    Caractéristique : Calcul de l'entropie ou vérification de distributions inhabituelles de caractères.

13. **Position du TLD**

    Description : Certaines URLs de phishing placent le TLD plus tôt dans l'URL pour tromper les utilisateurs.

Caractéristique : Vérification si le TLD apparaît dans une position inhabituelle dans la chaîne de l'URL.

14. **Présence de chemin, paramètres de requête et fragments**

    Description : Les chemins longs et complexes, les chaînes de requête ou les identifiants de fragments (#) peuvent indiquer une tentative de phishing.

    Caractéristique : Longueur et complexité des chemins et des paramètres de requête.

15. **Domaines ou mots-clés suspects**

    Description : Les URLs de phishing imitent souvent des marques ou des domaines populaires (par exemple, *"facebok.com"* au lieu de *"facebook.com"*).

    Caractéristique : Vérification des similitudes avec des noms de marques bien connus en utilisant des algorithmes de correspondance floue.

5.4 Etapes de mise en oeuvre

5.4.1 Importation et pretraitement des données

Cette étape comprend l'importation des bibliothèques nécessaires à l'analyse, ainsi que le chargement des données à partir du jeu de données. Elle inclut également la division des données en fonction de leurs caractéristiques et de la catégorie de l'URL, comme phishing, malware, ou légitime. Par ailleurs, un prétraitement est effectué pour préparer les données à l'apprentissage du modèle. Enfin, un processus d'indexation est appliqué, où les URLs légitimes sont assignées à l'index 0, tandis que les URLs malveillantes, telles que les malwares, reçoivent l'index 1.

```python
import pandas as pd
import re
import ipaddress
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split
from urllib.parse import urlparse
```

Listing 1 – Importation des bibliothèques

```python
df=pd.read_csv("malicious_phish.csv")
```

Listing 2 – Chargement des données

```python
df.head()
```

Listing 3 – Affichage des données

	url	type
0	br-icloud.com.br	phishing
1	mp3raid.com/music/krizz_kaliko.html	benign
2	bopsecrets.org/rexroth/cr/1.htm	benign
3	http://www.garage-pirenne.be/index.php?option=...	defacement
4	http://adventure-nicaragua.net/index.php?optio...	defacement

```
df=pd.read_csv("malicious_phish.csv")
```

Listing 4 – Chargement des données

Chargement et visualisation des données initiales.

```
# Create 'label' column based on the condition
df['type'] = df['type'].apply(lambda x: 0 if x == 'benign' else 1)
```

Listing 5 – Indexation des données

Indexation de la colonne type en fonction des labes des URL dans le cas d'un URL legitime le type correspondera a 1 dans le cas d'un URL suspect(phishing, malware...) le type recevera 1.

5.4.2 Extraction et preparation de notre jeu de données

Après l'importation du jeu de données contenant les URL et leur catégorie correspondante (par exemple, *légitime*, *phishing* ou *malware*), nous procédons à l'extraction des caractéristiques (*features*) à partir de chaque URL. Cette étape consiste à calculer une série de mesures spécifiques à partir des URL, telles que :

- **Longueur de l'URL** : Calcul de la taille totale de l'URL (nombre de caractères).
- **Nombre de points (`'.'`)** : Comptage du nombre de points présents dans l'URL.
- **Présence de sous-domaines** : Identification de la présence ou non de sous-domaines dans l'URL.
- **Utilisation de caractères spéciaux** : Vérification de la fréquence ou de la présence de caractères spéciaux comme `@`, `/`, ou `?`.

Ces caractéristiques sont calculées pour chaque URL et organisées dans un nouveau jeu de données. Ce nouveau dataset inclut les valeurs des caractéristiques extraites ainsi que la catégorie associée à chaque URL (par exemple, 0 pour *légitime* et 1 pour *malware*).

Ce dataset structuré sera ensuite utilisé comme entrée pour l'implémentation du modèle d'apprentissage automatique, qui apprendra à distinguer les différentes catégories d'URL en fonction des patterns observés dans les caractéristiques.

```python
def featureExtraction(url, label):
    features = []
    # Address bar-based features
    features.append(getDomain(url))                    # Domain name
    features.append(havingIP(url))                     # IP in URL
    features.append(haveAtSign(url))                   # '@' symbol
    features.append(getLength(url))                    # URL length
    features.append(getDepth(url))                     # Depth of URL
    features.append(useOfHTTPS(url))                   # HTTPS protocol
    features.append(isShortened(url))                  # URL shortener
        check
    features.append(countSubdomains(url))              # Subdomain count
    features.append(containsSuspiciousCharacters(url))  # Suspicious
        characters in URL
    features.append(domainLength(url))                 # Domain length
    features.append(digitCount(url))                   # Number of digits
        in URL
    features.append(containsSensitiveWords(url))       # Sensitive words
        in URL
    features.append(usesURLEncoding(url))              # URL encoding
        presence
    features.append(isSuspiciousTLD(url))              # Suspicious TLD
    features.append(httpsInDomain(url))                # "https" in domain
    features.append(hasRedirect(url))                  # Redirection in
        URL
    features.append(getPathLength(url))                # Path length

    # Include the label as the last feature
    features.append(label)

    return features
```

Listing 6 – Extraction des parametres de notre dataset

```
#converting the list to dataframe
feature_names = [
    'Domain',                 # getDomain(url)
    'Have_IP',                # havingIP(url)
    'Have_At',                # haveAtSign(url)
    'URL_Length',             # getLength(url)
    'URL_Depth',              # getDepth(url)
    'Use_of_HTTPS',           # useOfHTTPS(url)
    'TinyURL',                # isShortened(url)
    'Subdomain_Count',        # countSubdomains(url)
    'Suspicious_Characters',# containsSuspiciousCharacters(url)
    'Domain_Length',          # domainLength(url)
    'Digit_Count',            # digitCount(url)
    'Sensitive_Words',        # containsSensitiveWords(url)
    'URL_Encoding',           # usesURLEncoding(url)
    'Suspicious_TLD',         # isSuspiciousTLD(url)
    'HTTPS_in_Domain',        # httpsInDomain(url)
    'Redirection',            # hasRedirect(url)
    'Path_Length',            # getPathLength(url)
    'Label'                   # label
]
full_features_df = pd.DataFrame(features, columns=feature_names)
full_features_df.head()
```

Listing 7 – Création de la nouvelle Dataset

	Domain	Have_IP	Have_At	URL_Length	URL_Depth	Use_of_HTTPS	TinyURL	Subdomain_Count	Suspicious_Characters	Domain_Length	Digit_Count	Sensitive_Words	URL_Encoding	Suspicious_TLD	HTTPS_in_Domain	Redirection	Path_Length	Label
0		0	0	16	0	0	0	0	1	0	0	0	0	0	0	0	16	1
1		0	0	35	2	0	0	0	1	0	1	0	0	0	0	0	35	0
2		0	0	31	3	0	0	0	0	0	1	0	0	0	0	0	31	0
3	garage-pirenne.be	0	0	88	1	0	0	1	1	17	7	0	0	0	0	0	10	1
4	adventure-nicaragua.net	0	0	235	1	0	0	1	1	23	22	0	0	0	0	0	10	1

```
1 data.info()
```

Listing 8 – Nouveau jeu de données

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45287 entries, 0 to 45286
Data columns (total 18 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Domain                 13168 non-null  object
 1   Have_IP                45287 non-null  int64
 2   Have_At                45287 non-null  int64
 3   URL_Length             45287 non-null  int64
 4   URL_Depth              45287 non-null  int64
 5   Use_of_HTTPS           45287 non-null  int64
 6   TinyURL                45287 non-null  int64
 7   Subdomain_Count        45287 non-null  int64
 8   Suspicious_Characters  45287 non-null  int64
 9   Domain_Length          45287 non-null  int64
 10  Digit_Count            45287 non-null  int64
 11  Sensitive_Words        45287 non-null  int64
 12  URL_Encoding           45287 non-null  int64
 13  Suspicious_TLD         45287 non-null  int64
 14  HTTPS_in_Domain        45287 non-null  int64
 15  Redirection            45287 non-null  int64
 16  Path_Length            45287 non-null  int64
 17  Label                  45287 non-null  int64
dtypes: int64(17), object(1)
memory usage: 6.2+ MB
```

```
# Sepratating & assigning features and target columns to X & y
y = data1['Label']
X = data1.drop('Label',axis=1)
X.shape, y.shape
```

Listing 9 – Distribution des données

```
# Splitting the dataset into train and test sets: 80-20 split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
    random_state=12)
X_train.shape, X_test.shape
```

Listing 10 – Distribution des données

Dans cette section, nous évaluons les performances du modèle Naïve Bayes (`GaussianNB`) entraîné sur les données d'apprentissage et testé sur les données de test. Le code utilisé est présenté ci-dessous :

```
# Initialize the Na ve Bayes classifier
nb_model = GaussianNB()

# Train the model on the training data
nb_model.fit(X_train, y_train)

# Predict on the test data
y_pred = nb_model.predict(X_test)

# Print accuracy and classification report
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

Listing 11 – Entraînement et évaluation du modèle Naïve Bayes

Les performances du modèle sont évaluées à l'aide de la précision globale (*accuracy*) et d'un rapport de classification détaillé. Les résultats obtenus sont les suivants :

— **Précision globale (Accuracy)** : 0.89 (soit 89%). Cela signifie que le modèle a correctement classé 89% des URL testées.

Rapport de classification :

Classe	Précision	Rappel	F1-Score	Support
0 (légitime)	0.90	0.96	0.93	6693
1 (malware)	0.85	0.70	0.77	2365

TABLE 8 – Rapport de classification du modèle Naïve Bayes

Analyse des résultats :

- **Classe 0 (légitime)** : Le modèle montre une excellente performance pour détecter les URL légitimes, avec une précision de 90% et un rappel de 96%.
- **Classe 1 (malware)** : Bien que la précision soit de 85%, le rappel est plus faible (70%), indiquant que le modèle manque certaines URL malveillantes.

Pour améliorer les performances sur les URL malveillantes, il serait utile d'appliquer des techniques pour gérer le déséquilibre des classes, comme le sur-échantillonnage de la classe minoritaire ou l'ajustement des poids dans le modèle.

```
from sklearn.metrics import confusion_matrix

# Matrice de confusion
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
import matplotlib.pyplot as plt
import seaborn as sns

# Affichage graphique de la matrice de confusion
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['
    L gitime', 'Phishing'], yticklabels=['L gitime', 'Phishing'])

# Titres et labels
plt.title("Matrice de Confusion", fontsize=16)
plt.xlabel("Pr diction", fontsize=12)
```

```
17 plt.ylabel("R e l", fontsize=12)
18 plt.show()
```

Listing 12 – Evaluation a travers la matrice de confusion

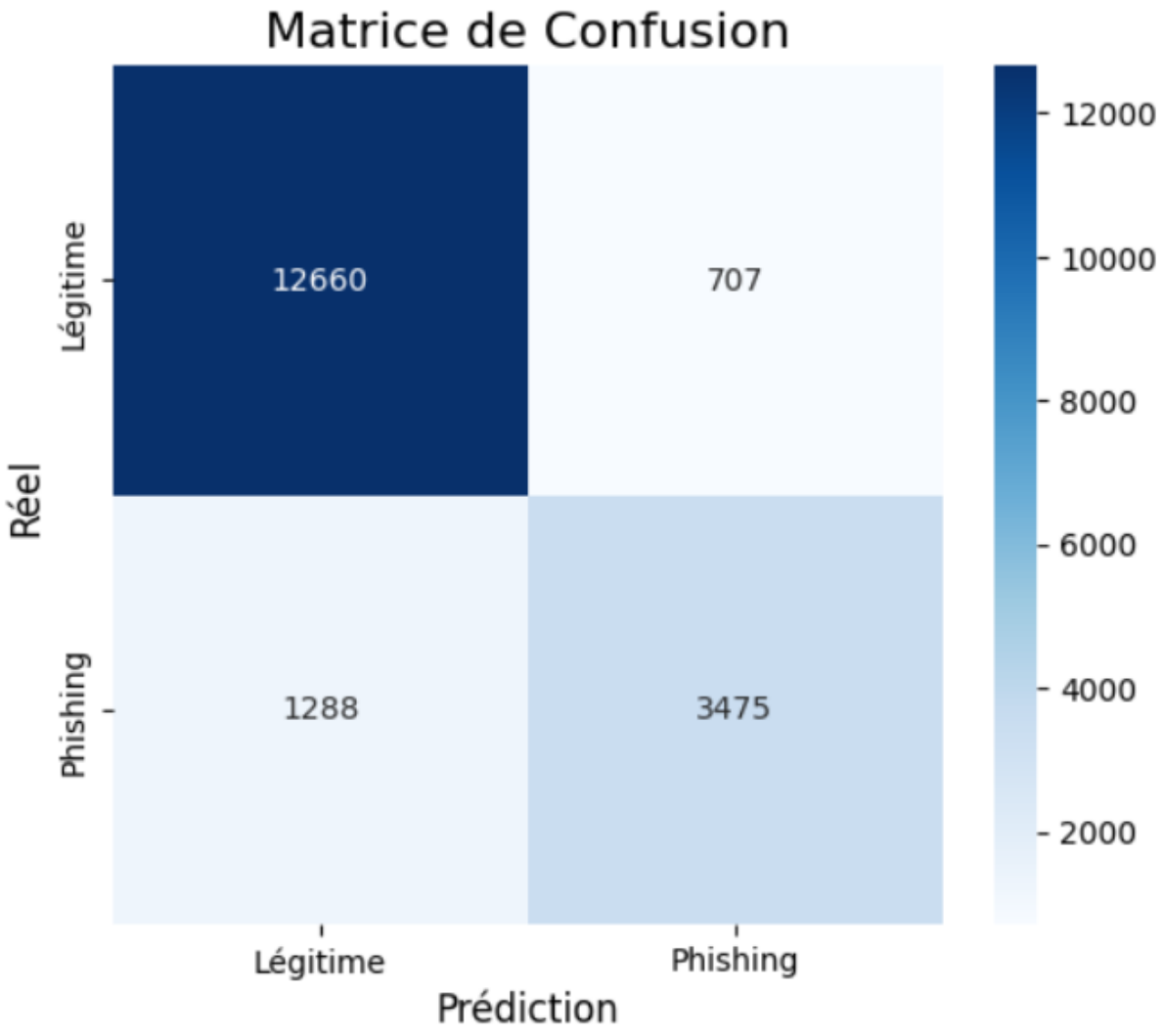

FIGURE 1 – Matrice de confusion pour le modèle Naïve Bayes.

- **Précision globale (Accuracy)** : 89.5%
- **Classe légitime (0)** :
  - **Précision** : 90.7%
  - **Rappel** : 94.7%
  - **F1-Score** : 92.6%

— **Classe phishing (1)** :
  — **Précision** : 83.1%
  — **Rappel** : 73.0%
  — **F1-Score** : 77.7%

Forces et Faiblesses

— **Forces :** Le modèle est très performant pour détecter les URLs légitimes, avec un rappel élevé de 94.7% et une précision de 90.7%.
— **Faiblesses :** Le modèle montre des lacunes pour détecter les URLs de phishing, avec un rappel relativement faible de 73.0%, ce qui laisse passer environ 27% des cas de phishing comme légitimes.

Améliorations possibles

— Ajuster les priorités des classes dans le modèle pour accorder plus d'importance à la détection de la classe *Phishing*.
— Collecter davantage de données pour la classe phishing afin de mieux entraîner le modèle.
— Améliorer l'extraction des caractéristiques pour rendre les URLs de phishing plus discriminantes par rapport aux URLs légitimes.

5.4.3 Test du Modèle à l'aide d'URLs Simulées

Dans cette section, nous testons la performance de notre modèle en introduisant un ensemble d'URLs simulées, représentatives de cas réels. Ces URLs, définies manuellement, incluent des exemples potentiels de sites légitimes et suspects, afin d'évaluer la capacité du modèle à différencier entre les deux catégories : *Légitime* et *Phishing*.

Procédure de Test

Le test suit les étapes suivantes :

1. Un ensemble d'URLs est déclaré manuellement. Ces URLs couvrent à la fois des sites couramment visités (par exemple, *facebook.com*) et des exemples plus inhabituels ou suspects (par exemple, *f051.9.1.41.xsph.ru*).
2. Chaque URL est analysée à l'aide de la fonction `featureExtraction`, qui génère un vecteur de caractéristiques spécifiques à partir de l'URL. Les caractéristiques utilisées incluent des mesures telles que la longueur de l'URL, la présence de sous-domaines, ou l'utilisation de mots sensibles.
3. Les vecteurs de caractéristiques sont stockés dans un DataFrame, avec les mêmes colonnes que les données utilisées pour entraîner le modèle. Toute colonne contenant des données non numériques est encodée à l'aide de `LabelEncoder`.
4. Le modèle Naïve Bayes (`nb_model`) préalablement entraîné est utilisé pour effectuer des prédictions sur cet ensemble de test. Le modèle retourne une prédiction pour chaque URL, indiquant si elle est *Légitime* (0) ou *Phishing* (1).

```
# D finir un ensemble d'URLs    tester
test_urls = [
    "https://www.facebook.com/",
    "https://www.funzine.hu",
    "https://chatgpt.com/",
    "https://www.bulgariaski.com",
    "https://www.aoh61.com",
    "https://www.southbankmosaics.com",
    "https://www.uni-mainz.de",
    "https://www.rewildingaaaaaaaaargentina.org",
    "http://www.teramill.com",
    "http://www.f051.9.1.41.xsph.ru"
]

# Extraire les caract ristiques pour chaque URL
test_features = []
for url in test_urls:
```

```
    features = featureExtraction(url, label=0)  # Label peut  tre
        ignor
    test_features.append(features[1:-1])        # Exclure le label pour
        le test

# Cr er un DataFrame    partir des caract ristiques
test_df = pd.DataFrame(test_features, columns=X.columns)

# Faire des pr dictions avec le mod le Na ve Bayes
predictions = nb_model.predict(test_df)

# Afficher les r sultats
for url, prediction in zip(test_urls, predictions):
    result = "Legitimate" if prediction == 0 else "Phishing"
    print(f"URL: {url} - Prediction: {result}")
```

Listing 13 – Test du modèle avec des URLs simulées

Résultats

Les résultats des prédictions sont affichés sous la forme d'un tableau associant chaque URL testée à sa classification par le modèle (*Légitime* ou *Phishing*). Cela permet d'évaluer visuellement les performances du modèle et d'identifier les cas où il peut se tromper.

```
URL: https://www.facebook.com/ - Prediction: Legitimate
URL: https://www.funzine.hu - Prediction: Legitimate
URL: https://chatgpt.com/ - Prediction: Legitimate
URL: https://www.bulgariaski.com - Prediction: Legitimate
URL: https://www.aoh61.com - Prediction: Legitimate
URL: https://www.southbankmosaics.com - Prediction: Legitimate
URL: https://www.uni-mainz.de - Prediction: Legitimate
URL: https://www.rewildingaaaaaaaaargentina.org - Prediction: Legitimate
URL: http://www.teramill.com - Prediction: Phishing
URL: http://www.f051.9.1.41.xsph.ru - Prediction: Phishing
URL: https://www.f051.9.1.41.xsph.ru - Prediction: Legitimate
```

Ce test met en évidence la capacité du modèle à généraliser sur des données réelles non vues lors de l'entraînement. En identifiant les erreurs (faux positifs ou faux négatifs), nous pouvons analyser les limites du modèle et envisager des améliorations telles que l'optimisation des caractéristiques ou l'ajustement des paramètres.

6 Conclusion Générale

Dans le cadre de ce projet, nous avons développé un modèle basé sur le classifieur Naïve Bayes pour la détection d'URLs de phishing, un enjeu crucial dans la lutte contre les cyberattaques. Le modèle a été entraîné à l'aide d'un ensemble de données comportant diverses caractéristiques discriminantes extraites des URLs, telles que la longueur de l'URL, le nombre de sous-domaines, l'utilisation de mots sensibles, ou encore la présence de caractères suspects.

Résumé des Étapes

Les étapes clés de ce projet incluent :

— **Prétraitement des données** : La création d'un ensemble de caractéristiques basé sur des observations structurelles des URLs a permis de fournir au modèle des informations pertinentes et exploitables. Une attention particulière a été accordée à l'encodage des colonnes non numériques pour garantir la compatibilité avec le modèle.

— **Modélisation avec Naïve Bayes** : Le choix de Naïve Bayes repose sur sa simplicité et son efficacité dans les problèmes de classification supervisée. Malgré son hypothèse forte d'indépendance entre les caractéristiques, le modèle a montré des performances satisfaisantes dans la détection des URLs légitimes et malveillantes.

— **Évaluation des performances** : Une matrice de confusion a permis de révéler les forces et limites du modèle. Avec une précision globale de 89.5%, le modèle détecte correctement la majorité des URLs légitimes, mais présente encore des faiblesses dans la détection des cas de phishing, avec un rappel de 73% pour cette classe.

Lors des tests sur des données simulées (URLs définies manuellement), le modèle a pu classifier les URLs comme *Phishing* ou *Légitime*. Ces résultats ont démontré une certaine capacité de généralisation du modèle, mais également la nécessité d'améliorer la détection des cas plus complexes ou ambigus.

Perspectives et Améliorations

- **Optimisation des caractéristiques** : L'ajout de nouvelles caractéristiques, notamment celles liées à des données dynamiques (comme l'analyse des pages web associées aux URLs), pourrait renforcer la capacité du modèle à différencier les URLs légitimes et malveillantes.
- **Gestion des déséquilibres de classe** : Bien que les classes soient relativement équilibrées, des techniques comme le sur-échantillonnage de la classe minoritaire ou l'ajustement des priorités dans le modèle pourraient réduire les faux négatifs pour la classe *Phishing*.
- **Extension à des modèles avancés** : Bien que Naïve Bayes soit un choix judicieux pour un problème d'entrée, des modèles plus avancés comme les forêts aléatoires, le gradient boosting ou même des réseaux neuronaux pourraient être explorés pour améliorer les performances globales.
- **Mise en œuvre en conditions réelles** : Enfin, l'intégration de ce modèle dans un système en temps réel pour analyser les URLs pourrait être envisagée. Cela nécessiterait des tests supplémentaires sur des données variées et non étiquetées pour évaluer la robustesse du modèle face à des scénarios du monde réel.

Conclusion Finale

En conclusion, ce projet a démontré la faisabilité et la pertinence de l'utilisation d'un modèle Naïve Bayes pour la détection d'URLs de phishing. Bien que les résultats soient encourageants, notamment pour la détection des URLs légitimes, des améliorations sont nécessaires pour mieux gérer les faux négatifs et renforcer la détection des cas de phishing. Ces travaux constituent une base solide pour des recherches futures et des développements pratiques visant à renforcer la cybersécurité à l'échelle individuelle et organisationnelle.