IV.3 L'application du Machine Learning pour la détection les URLs malicieux :

- **Dataset :**

L'ensemble d'URL de phishing est collecté à partir d'un service open source appelé PhishTank. Ce service fournit un ensemble d'URL de phishing dans plusieurs formats tels que csv, json, etc. qui sont mis à jour toutes les heures. Pour télécharger les données : .

Les URL légitimes sont obtenues à partir des ensembles de données ouverts de l'Université du Nouveau-Brunswick,

https://www.unb.ca/cic/datasets/url-2016.html.

Cet ensemble de données contient une collection d'URL bénignes, de spam, de phishing, de logiciels malveillants et de dégradation. Parmi tous ces types, l'ensemble de données d'URL bénigne est pris en compte pour ce projet.

La figure IV.9 décrit toutes les étapes impliquées dans le processus de gestion des ensembles de données ; du téléchargement des données à leur intégration dans un modèle de Machine Learning. Les principales étapes impliquées étaient les suivantes :

Prétraitement des données : suppression des valeurs nulles et des enregistrements en double car ils affectent notre modèle d'apprentissage automatique (ML). En outre, suppression des colonnes inutiles et conservé uniquement les colonnes URL et Label.

Les données prétraitées ont été vérifiées pour l'accessibilité de 'whois' pour s'assurer que nos données de fonctionnalités ne sont pas remplies de fonctionnalités insignifiantes et ne déséquilibrent pas le modèle ML.

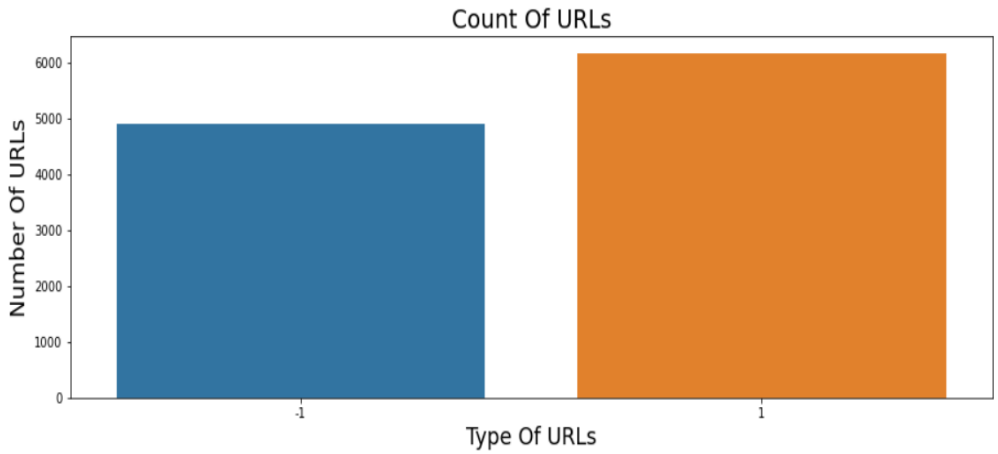

Figure IV.8: Tailles de données légitimes et du phishing (-1 : legitime,1 : Phishing).

Extraction de fonctionnalités : les URL bénignes et malveillantes ont été fusionnées pour obtenir un ensemble de données final permettant d'extraire les fonctionnalités. Les fonctionnalités ont été extraites.

Apprentissage automatique : les fonctionnalités extraites ont été utilisées pour former notre modèle ML et préparer notre machine pour l'URL classification.

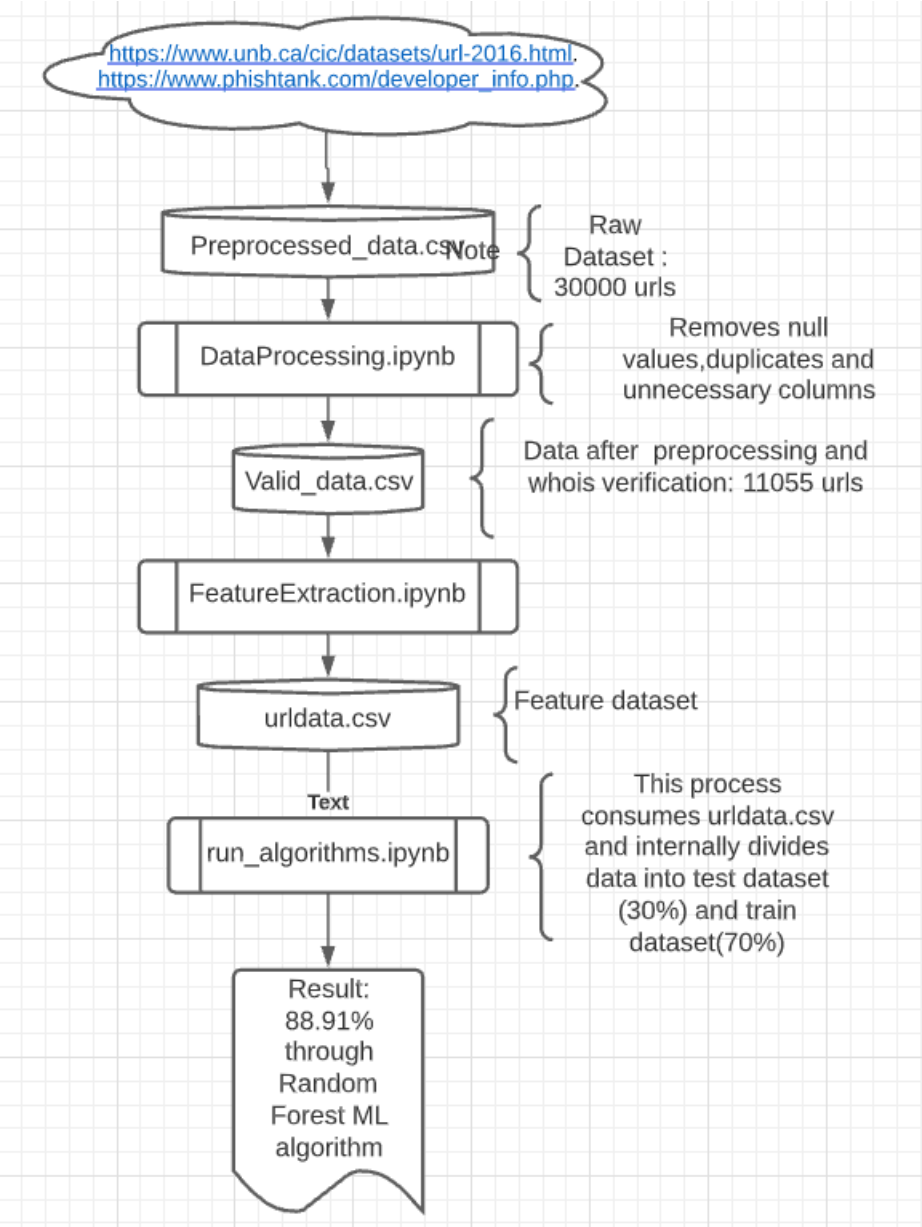

Figure IV.9: Dataset management process.

L'ensemble des données après l'extraction des fonctionnalités se présente comme suit :

	uses_ip	long_url	short_utl	@ symbol	redirect	hypen	favicon	https_token	req_url	anchor	linksintags	mailto	iframe	Label
0	1	1	1	1	1	-1	1	-1	1	0	-1	1	1	-1
1	1	0	1	1	1	-1	1	-1	1	0	-1	-1	1	-1
2	1	0	1	1	1	-1	1	-1	-1	0	0	1	1	-1
3	1	0	-1	1	1	-1	1	1	1	0	0	1	1	1
4	-1	0	-1	1	-1	-1	1	-1	1	0	0	-1	1	1

Figure IV.10: Dataset après l'extraction des fonctionnalités.

- **Extraction des fonctionalités:**

L'un des défis rencontrés par notre recherche était l'indisponibilité d'ensembles de données d'entraînement fiables. En fait, ce dséfi est confronté à tout chercheur dans le domaine. Cependant, bien que de nombreux articles sur la prédiction des sites Web de phishing à l'aide de techniques d'exploration de données aient été diffusés ces jours-ci, aucun ensemble de données de formation fiable n'a été publié publiquement, peut-être parce qu'il n'y a pas d'accord dans la littérature sur les caractéristiques définitives qui caractérisent les sites Web de phishing, il est donc difficile pour façonner un jeu de données qui couvre toutes les caractéristiques possibles. Dans cette section, nous mettons en lumière les fonctionnalités importantes qui se sont avérées solides et efficaces pour prédire les sites Web de phishing.

☐ **Utilisation de l'adresse IP:**

Si une adresse IP est utilisée comme alternative au nom de domaine dans l'URL, telle que "http://125.98.3.123/fake.html", les utilisateurs peuvent être sûrs que quelqu'un essaie de voler leurs informations personnelles. Parfois, l'adresse IP est même transformée en code hexadécimal comme indiqué dans le lien suivant

« http://0x58.0xCC.0xCA.0x62/2/paypal.ca/index.html ».

Règle :
SI URL Contient adresse IP →Phishing
SINON →Légitime

```
def having_ip_address(url):
    ip_address_pattern = ipv4_pattern + "|" + ipv6_pattern
    match = re.search(ip_address_pattern, url)
    return -1 if match else 1
```

Figure IV.11: extraction de la fonctionnalité « utilisation IP ».

☐ **URLs longues:**

Les hameçonneurs peuvent utiliser une URL longue pour masquer la partie douteuse dans la barre d'adresse. Par exemple:

http://federmacedoadv.com.br/3f/aze/ab51e2e319e51502f416dbe46b773a5e/?cmd=_home&dispatch=11004d58f5b74f8dc1e7c2e8dd4105e811004d58f5b74f8dc1e7c2e8dd4105e8@phishing.website.html.

Pour garantir l'exactitude de notre étude, nous avons calculé la longueur des URL dans l'ensemble de données et produit une longueur d'URL moyenne. Les résultats ont montré que si la longueur de l'URL est supérieure ou égale à 54 caractères, alors l'URL est classée comme hameçonnage. En examinant notre ensemble de données, nous avons pu trouver 1220 longueurs d'URL égales à 54 ou plus, ce qui constitue 48,8% de la taille totale de l'ensemble de données.

Règle :

SI url length <54 →Légitime

SINON 54<url length<75→Peut être

SINON →Phishing

Nous avons pu mettre à jour cette règle de fonctionnalité en utilisant une méthode basée sur la fréquence et en améliorant ainsi sa précision.

```
def long_url(url):
    if len(url) < 54:
        return 1
    if 54 <= len(url) <= 75:
        return 0
    return -1
```

Figure IV.12: extraction de la fonctionnalité « Long url ».

☐ **URLs courtes "TinyURL":**

Le raccourcissement d'URL est une méthode sur le « World Wide Web » dans laquelle une URL peut être considérablement réduite en longueur tout en menant à la page Web requise. Ceci est accompli au moyen d'une « redirection HTTP » sur un nom de domaine court, qui renvoie à la page Web qui a une longue URL. Par exemple, l'URL "http://portal.hud.ac.uk/" peut être raccourcie en "bit.ly/19DXSk4".

Règle :

SI URL courte →Phishing

SINON →Légitime

```
def shortening_service(url):
    match = re.search(shortening_services, url)
    return -1 if match else 1
```

Figure IV.13: extraction de la fonctionnalité « short_url ».

☐ **URLs avec le symbole "@" :**

L'utilisation du symbole « @ » dans l'URL conduit le navigateur à ignorer tout ce qui précède le symbole « @ » et l'adresse réelle suit souvent le symbole « @ ».

Règle :

SI Contient '@' →Phishing

SINON →Légitime

```python
def having_at_symbol(url):
    match = re.search('@', url)
    return -1 if match else 1
```

Figure IV.14: extraction de la fonctionnalité « @ Symbol ».

☐ **Redirection avec"//":**

L'existence de « // » dans le chemin de l'URL signifie que l'utilisateur sera redirigé vers un autre site Web. Un exemple de telles URL est : "http://www.legitimate.com//http://www.phishing.com". Nous examinons l'emplacement où le « // » apparaît. Nous constatons que si l'URL commence par "HTTP", cela signifie que le "//" doit apparaître en sixième position. Cependant, si l'URL utilise « HTTPS », alors le « // » devrait apparaître en septième position.

Règle :

SI Position de la dernière occurrence '//'>7 →Phishing

SINON→Légitime

```python
def double_slash_redirecting(url):
    # Comme la position commence de 0, la position du dernier // doit etre >6.
    last_double_slash = url.rfind('//')
    return -1 if last_double_slash > 6 else 1
```

Figure IV.15: extraction de la fonctionnalité « redirect // ».

☐ **URLs avec "-":**

Le symbole tiret est rarement utilisé dans les URL légitimes. Les hameçonneurs ont tendance à ajouter des préfixes ou des suffixes séparés par (-) au nom de domaine afin que les utilisateurs sentent qu'ils ont affaire à une page Web légitime. Par exemple : http://www.Confirme-paypal.

Règle :

SI URL Contient le symbole'-' → Phishing

SINON→Légitime

```python
def is_hypen(domain):
    match = re.search('-', domain)
    return -1 if match else 1
```

Figure IV.16: extraction de la fonctionnalité « hypen ».

☐ **Favicon:**

Un favicon est une image graphique (icône) associée à une page Web spécifique. De nombreux agents utilisateurs existants, tels que les navigateurs graphiques et les lecteurs de nouvelles, affichent le favicon comme un rappel visuel de l'identité du site Web dans la barre d'adresse. Si le favicon est chargé à partir d'un domaine autre que celui affiché dans la barre d'adresse, la page Web est susceptible d'être considérée comme une tentative de phishing.

Règle :

Favicon d'un domaine externe→Phishing

SINON→Légitime

```
def favicon(url):
    domain = get_hostname_from_url(url)
    req = requests.get(url)
    soup = BeautifulSoup(req.content, 'html.parser')
    for head in soup.find_all('head'):
        for head.link in soup.find_all('link', href=True):
            dots = [x.start() for x in re.finditer(r'\.', head.link['href'])]
            return 1 if url in head.link['href'] or len(dots) == 1 or domain in head.link['href'] else -1
    return 1
```

Figure IV.17: extraction de la fonctionnalité « favicon ».

☐ **'HTTPS' dans la partie 'Domain' de l'url :**

Les hameçonneurs peuvent ajouter le jeton « HTTPS » à la partie domaine d'une URL afin de tromper les utilisateurs. Par exemple,

http://https-www-paypal-it-webapps-mpp-home.soft-hair.com/.

Règle :

SI HTTPS dans la partie Domain →Phishing

SINON→Légitime

```
def https_token(url):
    match = re.search(http_https, url)
    if match and match.start() == 0:
        url = url[match.end():]
    match = re.search('http|https', url)
    return -1 if match else 1
```

Figure IV.18: extraction de la fonctionnalité « https_token ».

☐ **Request URL:**

L'URL de demande examine si les objets externes contenus dans une page Web, tels que des images, des vidéos et des sons, sont chargés à partir d'un autre domaine. Dans les pages Web légitimes, l'adresse de la page Web et la plupart des objets intégrés à la page Web partagent le même domaine.

Règle :

SI % Request URL<22% →Légitime

SI 22%<Request URL<61% →Peut être

SINON→Phishing

```
def request_url(wiki, soup, domain):
    i = 0
    success = 0
    for img in soup.find_all('img', src=True):
        dots = [x.start() for x in re.finditer(r'\.', img['src'])]
        if wiki in img['src'] or domain in img['src'] or len(dots) == 1:
            success = success + 1
        i = i + 1

    for audio in soup.find_all('audio', src=True):
        dots = [x.start() for x in re.finditer(r'\.', audio['src'])]
        if wiki in audio['src'] or domain in audio['src'] or len(dots) == 1:
            success = success + 1
        i = i + 1

    for embed in soup.find_all('embed', src=True):
        dots = [x.start() for x in re.finditer(r'\.', embed['src'])]
        if wiki in embed['src'] or domain in embed['src'] or len(dots) == 1:
            success = success + 1
        i = i + 1

    try:
        percentage = success / float(i) * 100
    except:
        return 1

    if percentage < 22.0:
        return 1
    elif 22.0 <= percentage < 61.0:
        return 0
    else:
        return -1
```

Figure IV.19: extraction de la fonctionnalité « req_url ».

☐ **URL of Anchor :**

Une ancre est un élément défini par la balise <a>. Cette fonctionnalité est traitée exactement comme « URL de demande ». Cependant, pour cette fonctionnalité, nous examinons:

*Si les balises <a> et le site Web ont des noms de domaine différents. Ceci est similaire à la fonctionnalité d'URL de demande.

*Si l'ancre ne renvoie à aucune page Web, par exemple :

A. <a href="#">

B. <a href="#content">

C. <a href="#skip">

D. <a href="JavaScript ::void(0)">

Règle :

SI %Anchor URL<31% →Légitime

SI 31%<% Anchor URL<67% →Peut être

SINON→Phishing

```
def url_of_anchor(wiki, soup, domain):
    i = 0
    unsafe = 0
    for a in soup.find_all('a', href=True):

        if "#" in a['href'] or "javascript" in a['href'].lower() or "mailto" in a['href'].lower() or not (
                wiki in a['href'] or domain in a['href']):
            unsafe = unsafe + 1
        i = i + 1

    try:
        percentage = unsafe / float(i) * 100
    except:
        return 1
    if percentage < 31.0:
        return 1
    elif 31.0 <= percentage < 67.0:
        return 0
    else:
        return -1
```

Figure IV.20: extraction de la fonctionnalité « anchor ».

☐ **URLs dans les tags<Script>, <Links> et<Tags>:**

Étant donné que notre enquête couvre tous les angles susceptibles d'être utilisés dans le code source de la page Web, nous constatons qu'il est courant que des sites Web légitimes utilisent des balises <Script> pour créer un script côté client ; et les balises <Link> pour récupérer d'autres ressources Web. Il est prévu que ces balises soient liées au même domaine de la page Web.

Règle :

SI % URL dans <Script> et <Link> <17% →Légitime

SI 17%<% URL dans <Script> et <Link> ,> 81% →Peut être

SINON →Phishing

```
def links_in_tags(wiki, soup, domain):
    i = 0
    success = 0
    for link in soup.find_all('link', href=True):
        dots = [x.start() for x in re.finditer(r'\.', link['href'])]
        if wiki in link['href'] or domain in link['href'] or len(dots) == 1:
            success = success + 1
        i = i + 1

    for script in soup.find_all('script', src=True):
        dots = [x.start() for x in re.finditer(r'\.', script['src'])]
        if wiki in script['src'] or domain in script['src'] or len(dots) == 1:
            success = success + 1
        i = i + 1
    try:
        percentage = success / float(i) * 100
    except:
        return 1

    if percentage < 17.0:
        return 1
    elif 17.0 <= percentage < 81.0:
        return 0
    else:
        return -1
```

Figure IV.21: extraction de la fonctionnalité « linksintags ».

☐ **URLs avec “mailto”:**

Le formulaire Web permet à un utilisateur de soumettre ses informations personnelles qui sont dirigées vers un serveur pour traitement. Un hameçonneur peut rediriger les informations de l'utilisateur vers son e-mail personnel. À cette fin, un langage de script côté serveur peut être utilisé, tel que la fonction "mail ()" en PHP. Une autre fonction côté client qui pourrait être utilisée à cette fin est la fonction "mailto:".

Règle :

SI URL utilise “mailto”→Phishing

SINON →Légitime

```
def submitting_to_email(soup):
    for form in soup.find_all('form', action=True):
        return -1 if "mailto:" in form['action'] else 1
    # Si pas de form donc retourner 1.
    return 1
```

Figure IV.22: extraction de la fonctionnalité « mailto ».

☐ **Redirection IFrame :**

IFrame est une balise HTML utilisée pour afficher une page Web supplémentaire dans celle qui est actuellement affichée. Les hameçonneurs peuvent utiliser la balise « iframe » et la rendre invisible, c'est-à-dire sans bordures de cadre. À cet égard, les hameçonneurs utilisent l'attribut « frameBorder » qui amène le navigateur à afficher une délimitation visuelle.

Règle :

SI URL utilise IFrame →Phishing

SINON →Légitime

```
def i_frame(soup):
    for i_frame in soup.find_all('i_frame', width=True, height=True, frameBorder=True):
        # Si un des iframes satisfsait ces conditions, retourner -1.
        if i_frame['width'] == "0" and i_frame['height'] == "0" and i_frame['frameBorder'] == "0":
            return -1
        if i_frame['width'] == "0" or i_frame['height'] == "0" or i_frame['frameBorder'] == "0":
            return 0
    #  Si pas de iframes avec 0 pour longueur ou largeur ou frameBorder donc c'est legitime ,retourner 1.
    return 1
```

Figure IV.23: extraction de la fonctionnalité « iframe».

- **Application du modèle de l’apprentissage automatique:**

Vu que notre ensemble de données et structuré et étiqueté nous pouvons implémenter un algorithme d’apprentissage automatique pour détecter les urls malicieux. Mais nous devons d’abord choisir parmi plusieurs algorithmes d’apprentissage supervisé qui peuvent traiter les

données étiquetées. Nous choisissons plusieurs algorithmes de classifieur, puis nous allons choisir l'algorithme qui donne les meilleures performances comme solution au problème, en analysant les métriques comme le taux de précision et le taux d'erreur.

Nous appliquons les algorithmes suivants sur l'ensemble des données:

- **Multi-Layer Perceptron:**

  Le perceptron multicouche (MLP) est un complément du réseau de neurones à action directe. Il se compose de trois types de couches : la couche d'entrée, la couche de sortie et la couche cachée, comme le montre la figure IV.24. La couche d'entrée reçoit le signal d'entrée à traiter. La tâche requise telle que la prédiction et la classification est effectuée par la couche de sortie. Un nombre arbitraire de couches cachées qui sont placées entre la couche d'entrée et la couche de sortie sont le véritable moteur de calcul du MLP. Semblable à un réseau d'alimentation directe dans un MLP, les données circulent dans le sens direct de la couche d'entrée à la couche de sortie. Les neurones du MLP sont entraînés avec l'algorithme d'apprentissage par propagation arrière. Les MLP sont conçus pour approcher n'importe quelle fonction continue et peuvent résoudre des problèmes qui ne sont pas linéairement séparables. Les principaux cas d'utilisation de MLP sont la classification, la reconnaissance, la prédiction et l'approximation de modèles.

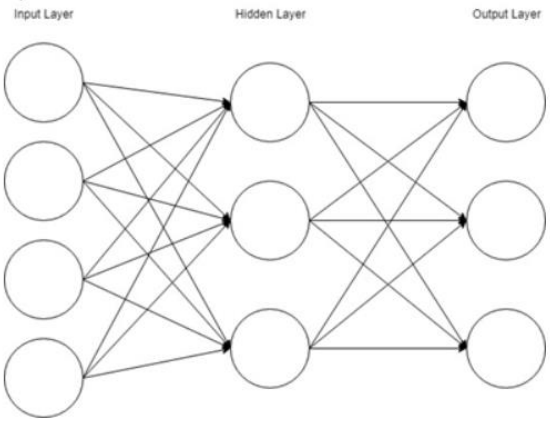

Figure IV.24: Multi-layer Perceptron.

L'algorithme du MLP est le suivant :

1. Tout comme avec le perceptron, les entrées sont poussées vers l'avant à travers le MLP en prenant le produit scalaire de l'entrée avec les poids qui existent entre la couche d'entrée et la couche cachée (WH). Ce produit scalaire donne une valeur au niveau de la couche cachée. Cependant, nous ne poussons pas cette valeur en avant comme nous le ferions avec un perceptron.
2. Les MLP utilisent des fonctions d'activation à chacune de leurs couches calculées. Il existe de nombreuses fonctions d'activation à discuter : unités linéaires rectifiées (ReLU), fonction sigmoïde, tanh. Poussez la sortie calculée au niveau de la couche actuelle via l'une de ces fonctions d'activation.
3. Une fois que la sortie calculée au niveau de la couche cachée a été transmise à travers la fonction d'activation, poussez-la vers la couche suivante dans le MLP en prenant le produit scalaire avec les poids correspondants. Répétez les étapes deux et trois jusqu'à ce que la couche de sortie soit atteinte.
4. Au niveau de la couche de sortie, les calculs seront soit utilisés pour un algorithme de rétropropagation qui correspond à la fonction d'activation qui a été

sélectionnée pour le MLP (dans le cas de la formation), soit une décision sera prise en fonction de la sortie (dans le cas des tests).

Les MLP constituent la base de tous les réseaux de neurones et ont considérablement amélioré la puissance des ordinateurs lorsqu'ils sont appliqués aux problèmes de classification et de régression. Les ordinateurs ne sont plus limités par les cas XOR et peuvent apprendre des modèles riches et complexes grâce au perceptron multicouche

- **Random Forest:**

Random Forest est un algorithme d'apprentissage supervisé. La "forêt" qu'il construit, est un ensemble d'arbres de décision, généralement formés avec la méthode du "bagging". L'idée générale de la méthode d'ensachage est qu'une combinaison de modèles d'apprentissage augmente le résultat global. L'un des grands avantages de la forêt aléatoire est qu'elle peut être utilisée à la fois pour des problèmes de classification et de régression, qui constituent la majorité des systèmes d'apprentissage automatique actuels. Examinons la forêt aléatoire dans la classification, car la classification est parfois considérée comme la pierre angulaire de l'apprentissage automatique.

Ci-dessous, vous pouvez voir à quoi ressemblerait une forêt aléatoire avec deux arbres :

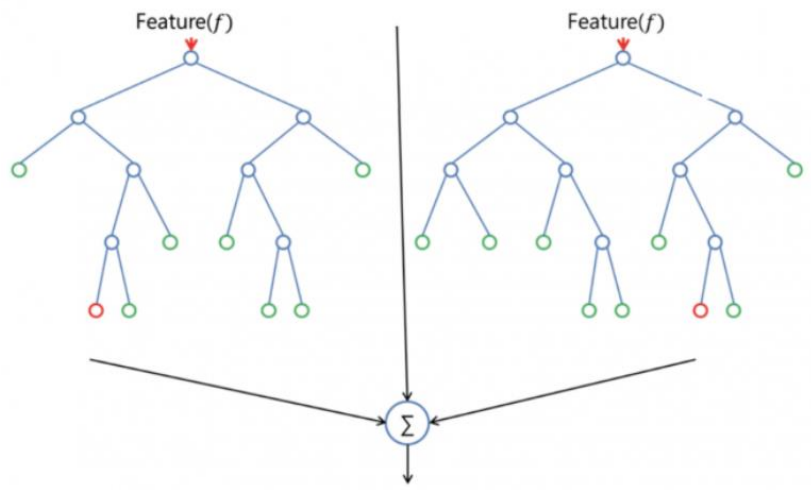

Figure IV.25: Random Forest avec deux arbres.

La forêt aléatoire a presque les mêmes hyperparamètres qu'un arbre de décision ou un classificateur d'ensachage. Heureusement, il n'est pas nécessaire de combiner un arbre de décision avec un classificateur d'ensachage car vous pouvez facilement utiliser la classe de classificateur de forêt aléatoire. Avec la forêt aléatoire, vous pouvez également gérer les tâches de régression en utilisant le régresseur de l'algorithme.

La forêt aléatoire ajoute un caractère aléatoire supplémentaire au modèle, tout en faisant pousser les arbres. Au lieu de rechercher la fonctionnalité la plus importante tout en divisant un nœud, il recherche la meilleure fonctionnalité parmi un sous-ensemble aléatoire de fonctionnalités. Il en résulte une grande diversité qui se traduit généralement par un meilleur modèle.

Une autre grande qualité de l'algorithme de forêt aléatoire est qu'il est très facile de mesurer l'importance relative de chaque caractéristique sur la prédiction. Sklearn fournit un excellent outil pour cela qui mesure l'importance d'une fonctionnalité en examinant dans quelle mesure les nœuds d'arbre qui utilisent cette fonctionnalité réduisent les impuretés dans tous les random forests. Il calcule ce score automatiquement pour chaque caractéristique après la formation et met à l'échelle les résultats de sorte que la somme de toutes les importances soit égale à un.
En examinant l'importance des fonctionnalités, nous pouvons décider quelles fonctionnalités à supprimer, car elles ne contribuent pas suffisamment (ou parfois rien du tout) au processus de prédiction. Ceci est important car une règle générale en apprentissage automatique est que plus nous avons de fonctionnalités, plus notre modèle risque de souffrir de « overfitting » et vice versa.

- **Support Vector Machine (SVM):**
  Une machine à vecteurs de support est un modèle d'apprentissage supervisé populaire développé par Vladimir Vapnik, utilisé à la fois pour la classification et la régression des données. Cela dit, il est généralement utilisé pour les problèmes de classification, en construisant un hyperplan où la distance entre deux classes de points de données est à son maximum. Cet hyperplan est connu sous le nom de frontière de décision, séparant les classes de points de données.

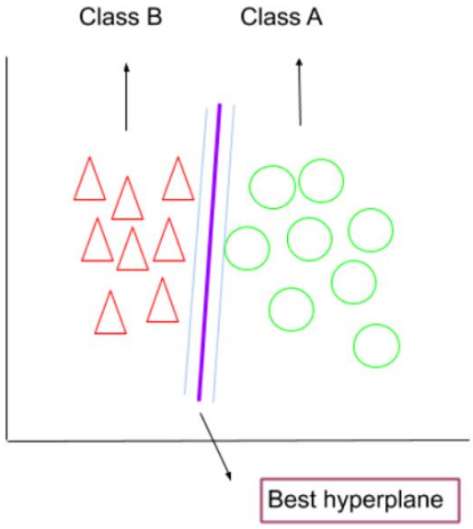

Figure IV.26: Random Forest avec deux arbres.

Pour appliquer les algorithmes sur l'ensemble de données, il faut diviser le dataset en deux, dataset d'entraînement et dataset d'évaluation. La figure ci-dessous montre le classifieur d'apprentissage supervisé.

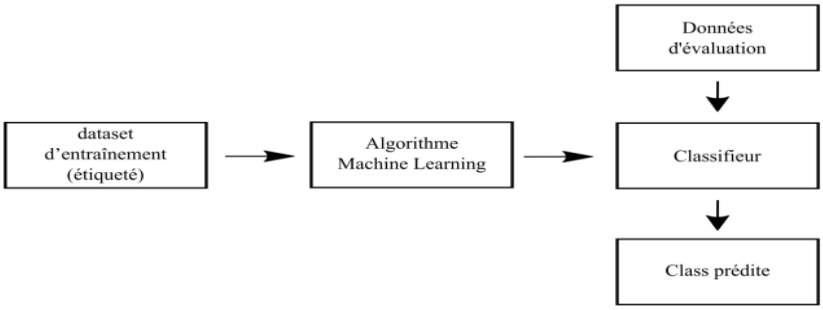

Figure IV.27: Classifeur d'apprentissage supervisé.

Ici, le classifieur applique le modèle choisi sur le dataset étiqueté pour l'entraînement, puis applique le modèle sur la portion du dataset dédiée pour l'évaluation des performances du modèle.

☐ **Phase d'entrainement :**

Avec un ensemble de données des fonctionnalités sélectionnées, nous pouvons maintenant former les modèles d'apprentissage automatique. La première étape consiste à obtenir l'ensemble de données d'entrée, puis à le traiter dans la deuxième étape.

L'étape suivante consiste à deviser l'ensemble en deux sous-ensembles, sous-ensemble d'entraînement qui forme 70% de l'ensemble global et le sous-ensemble d'évaluation qui forme 30% de l'ensemble global. Le sous-ensemble d'entrainement doit être plus grand que l'ensemble d'évaluation pour que le modèle s'entraîne avec une grande partie de l'ensemble de données et pour avoir des taux de précision élevée. Nous formons ensuite les modèles à l'aide des algorithmes d'apprentissage automatique provenant de bibliothèques Python scikit-learn.

☐ **Phase d'évaluation :**

Après que le modèle fini l'entraînement dans la phase d'entraînement, l'algorithme applique son apprentissage sur le sous-ensemble de test et affiche les résultats obtenus sous forme de matrice de confusion. À partir de la matrice de confusion nous pouvons extraire les informations relatives à la précision et le taux d'erreur.

Les performances de l'algorithme d'apprentissage automatique doivent être évaluées en utilisant les paramètres suivants : la précision, l'erreur et la précision. Nous utilisons une matrice de confusion pour calculer ces mesures de performance.

✦ ***Matrice de confusion :***

Une matrice de confusion est une matrice N × N où N est le nombre de classes. Dans ce mémoire, nous avons 2 classes (bénin et malveillant). Les colonnes de la matrice représentent les classes réelles et les lignes représentent les classes prédites. La matrice de confusion donne le total des résultats correctement et incorrectement prévus par le modèle.

TP = 'True Positive' (vrai positif) est le nombre de fois que les urls malicieux ont été correctement classé.

FN = 'False Negative' (faux négatif) est le nombre de fois que les urls malicieux ont été classé comme légitime.

TN = 'True Negative' (vrai négatif) est le nombre de prédictions correctement classées.

FP = 'False Positive' (faux positif) est le nombre de fois que les url légitimes ont été classé comme url malicieux.

À partir de la matrice de confusion, nous pouvons définir les mesures de performances suivantes : Taux de précision (Accuracy rate), Taux d'erreur (Error rate) et Précision (Precision).

Taux de précision :

Le taux de précision est donné par l'équation suivante :

Taux de précision = nombre de prédictions correctes / nombre total de prédictions.

Taux de précision = (TP + TN) / (TP + FN + FP + TN)

Taux d'erreur :

Le taux d'erreur est donné par l'équation suivante :

Taux d'erreur = nombre de prévisions erronées / nombre total de prévisions.

Taux d'erreur = (FP + FN) / (TP + FN + FP + TN)

Précision :

La précision est donnée par l'équation suivante :

Précision = nombre d'éléments pertinents sélectionnés.

Précision = TP/ (TP + FP)

- **Résultats:**

☐ **Multi-layer perceptron :**

```
accuracy = 88.09%
          Predicted 0  Predicted 1
Actual 0         1240          215
Actual 1          180         1682
0.8949188613993084
runtime = 6.2794835567474365 seconds
```

Figure IV.28: Résultats MLP.

☐ **Random Forest :**

```
accuracy = 88.91%
          Predicted 0  Predicted 1
Actual 0         1199          256
Actual 1          112         1750
0.9048603929679421
runtime = 0.0624845027923584 seconds
```

Figure IV.29: Résultats Random Forest.

☐ **SVM :**

```
accuracy = 86.49%
          Predicted 0  Predicted 1
Actual 0         1148          307
Actual 1          141         1721
0.8848329048843186
runtime = 1.746504783630371 seconds
runtime = 8.088472843170166 seconds
```

Figure IV.30: Résultats SVM.

Donc, le modèle à choisir est bien évidemment le 'Random Forest' car il a le taux de succès le plus élevé.

- **Création d'une extension de browser pour la détection des urls malicieux :**

L'extension Chrome de détection de phishing vise à classer, chaque URL parcourue, dans les catégories hameçonnées et non hameçonnées (au chargement de la page). ainsi, alerter l'utilisateur de toute activité malveillante et empêcher toute intrusion.

manifest.json : il fournit à Chrome les informations de base sur l'extension, telles que le nom, les autorisations, les scripts et les fichiers associés.

content.js : il s'exécute dans un environnement javscript non privilégié séparé et a un accès complet au DOM. Ici, le « modèle Random Forest » formé (poids calculés dans ./ML/Evaluation/run_algorithms.py) a été utilisé comme modèle persistant pour classer les sites Web.

Le cadre noir indique les lignes de code utilisées pour récupérer les coefficients des fonctionnalités

```python
def random_forests(dataset,class_labels,test_size):

    import numpy as np
    import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.ensemble import RandomForestClassifier
    from sklearn import metrics
    X = pd.read_csv(dataset)
    Y = pd.read_csv(class_labels)
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size= test_size, random_state=42)
    model = RandomForestClassifier(n_estimators = 5, criterion = 'entropy',random_state = 42)
    model.fit(X_train,y_train.values.ravel())
    Y_predicted = model.predict(X_test)
    importance = model.feature_importances_
    for i,v in enumerate(importance):
        print ('Feature: %0d, Score:%.5f' % (i,v))

    return y_test,Y_predicted
```

Figure IV.31: Récupération des coefficients des fonctionnalités.

Output:

```
Feature: 0, Score:0.01796
Feature: 1, Score:0.02181
Feature: 2, Score:0.01820
Feature: 3, Score:0.00986
Feature: 4, Score:0.01228
Feature: 5, Score:0.10514
Feature: 6, Score:0.01369
Feature: 7, Score:0.01414
Feature: 8, Score:0.04426
Feature: 9, Score:0.61717
Feature: 10, Score:0.09366
Feature: 11, Score:0.01178
Feature: 12, Score:0.01151
Feature: 13, Score:0.00853
```

Figure IV.32: Les coefficients des caractéristiques

Les fonctions ci-dessous calculent le vecteur de caractéristiques pour le portail en cours d'analyse :

- uses_ip()
- long_url()
- short_url()
- at_symbol()
- redirect_url()
- hypen()
- favicon()
- https_token()
- req_url()
- anchor()
- linksintags()
- mailto()
- iframe()

Le vecteur de caractéristiques évalué, en outre, transmis à la fonction predict(data) calcule la prédiction pour le site Web.

```
function predict(data,weight){
    var f = 0;
    weight = [0.01796,0.02181,0.01820,0.00986,0.01228,0.10514,0.01369,0.01414,0.04426,0.61717,0.09366,0.01178,0.01151,0.00853];
    for(var j=0;j<data.length;j++) {
      f += data[j] * weight[j];
    }
    return f > 0 ? 1 : -1;
}
```

Figure IV.33: Fonction predict(data).

- **Test de l'extension :**
  Site www.google.com

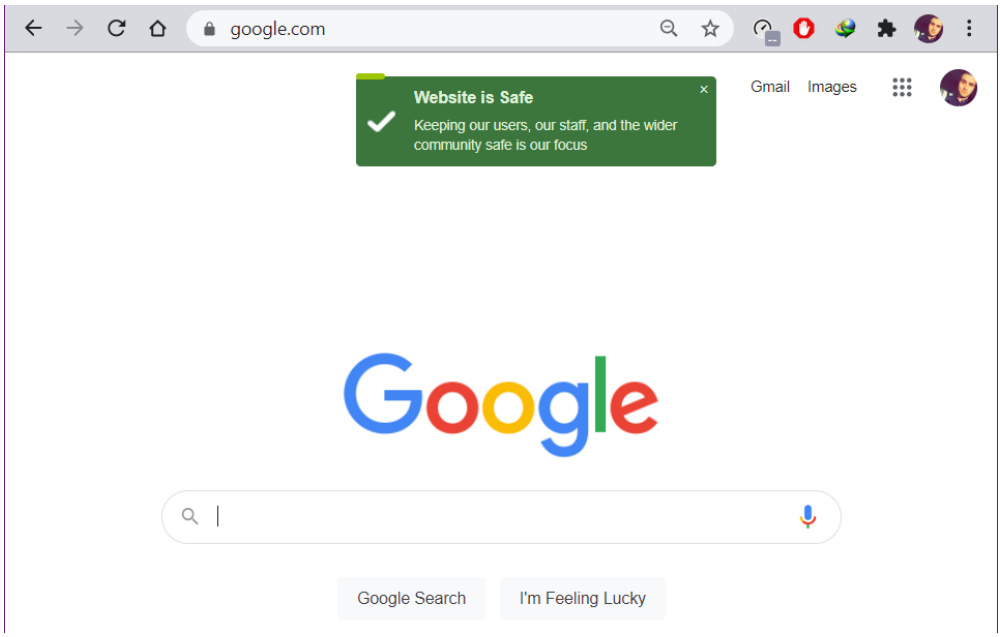

Figure IV.34: Test sur www.google.com

Le site http://cybercrime-tracker.net/ contient plusieurs liens de phishing pour nous aider à tester notre solution :

On choisit un lien parmi les plus récents vu que les urls de phishing ont un cycle de vie trop court :

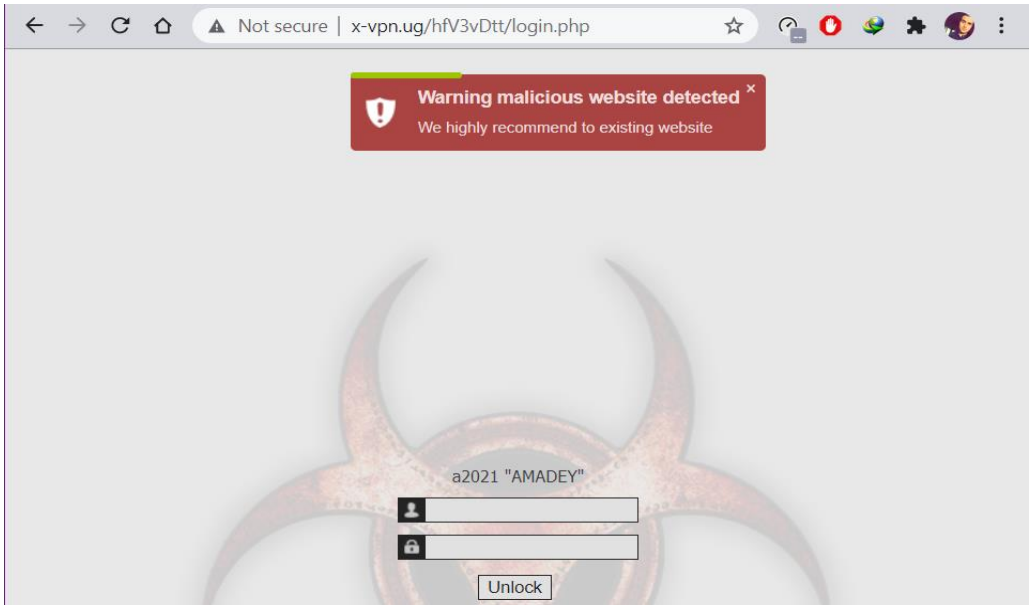

Figure IV.35: Test sur un site de Phishing.

Comme nous pouvons le constater l'extension a réussi de détecter le lien malicieux.

- **Technologies utilisées :**

☐ **Environnement :**

✦ **Python :**

Il existe différents langages de programmation qui peuvent être utilisés pour la science des données (par exemple, SQL, Java, Matlab, SAS, R et bien d'autres), mais Python est le choix préféré des data scientists parmi tous les autres langages de programmation de cette liste. Python a quelques fonctionnalités préférables extraordinaires, notamment :

- Python est très puissant et simple, il est donc facile d'apprendre le langage. Vous n'avez pas à vous soucier de sa syntaxe si vous êtes débutant.
- Python prend en charge de nombreuses plates-formes telles que Windows, Mac, Linux, etc.
- Python est un langage de programmation de haut niveau, vous écrivez donc un programme dans un proche anglais simple et celui-ci sera converti en interne en code de bas niveau.
- Python est un langage interprété qui signifie qu'il exécute le code une instruction à la fois.
- Python peut effectuer la visualisation de données, l'analyse de données et la manipulation de données ; NumPy et Pandas sont quelques-unes des bibliothèques utilisées pour la manipulation.
- Python sert diverses bibliothèques puissantes pour l'apprentissage automatique et les calculs scientifiques. Divers calculs scientifiques complexes et algorithmes d'apprentissage automatique peuvent être effectués à l'aide de ce langage facilement dans une syntaxe relativement simple.

☐ **Bibliothèques :**

✦ **NumPy :** NumPy est une bibliothèque open source disponible gratuitement en Python, qui signifie « Numerical Python ». C'est une bibliothèque Python populaire qui est utile dans les calculs scientifiques qui fournissent des objets de tableau, ainsi que des outils pour intégrer C et C++. NumPy fournit un puissant tableau à N dimensions qui se présente sous la forme de lignes et de colonnes. Ceux-ci peuvent être initialisés à partir d'une liste Python.

✦ **SciKit :** Cette bibliothèque populaire est utilisée pour l'apprentissage automatique en science des données avec divers algorithmes de classification, de régression et de clustering, qui fournit des machines à vecteurs de support, Bayes naïf, l'amplification de gradient et la régression logique.

✦ **Pandas :**

Pandas est connu pour fournir des trames de données en Python. Il s'agit d'une bibliothèque puissante pour l'analyse de données, par rapport à d'autres langages spécifiques à un domaine comme R. En utilisant Pandas, il est plus facile de gérer les données manquantes, prend en charge le travail avec des données indexées différemment recueillies à partir de plusieurs ressources différentes et prend en charge l'alignement automatique des données. Il fournit également des outils pour l'analyse des données et des structures de données telles que la fusion, la

mise en forme ou le découpage d'ensembles de données, et il est également très efficace pour travailler avec des données liées aux séries chronologiques en fournissant des outils robustes pour le chargement de données à partir d'Excel, de fichiers plats et de bases de données.

- **Matplotlib:** Matplotlib signifie 'Mathematical Plotting Library' en Python. Il s'agit d'une bibliothèque principalement utilisée pour la visualisation de données, notamment des tracés 3D, des histogrammes, des tracés d'images, des nuages de points, des graphiques à barres et des spectres de puissance avec des fonctions interactives de zoom et de panoramique pour la publication dans différents formats papier. Il prend en charge presque toutes les plates-formes telles que Windows, Mac et Linux. Cette bibliothèque sert également d'extension pour la bibliothèque NumPy. Matplotlib a un module pyplot qui est utilisé dans les visualisations, qui est souvent comparé à MATLAB.

- **Beautiful Soup :** Beautiful Soup est un package Python permettant d'analyser des documents HTML et XML (y compris des balises mal formées, c'est-à-dire des balises non fermées, ainsi nommées d'après la soupe de balises). Il crée un arbre d'analyse pour les pages analysées qui peut être utilisé pour extraire des données du HTML,ce qui est utile pour le Web scraping.

☐ **Outils :**

- **Jupyter Notebook :** Jupyter Notebook est une application Web open source qui vous permet de créer et de partager des documents contenant du code en direct, des équations, des visualisations et du texte narratif. Les utilisations incluent : le nettoyage et la transformation des données, la simulation numérique, la modélisation statistique, la visualisation des données, l'apprentissage automatique et bien plus encore.

- **Anaconda :** Anaconda est une distribution des langages de programmation Python et R pour le calcul scientifique (science des données, applications d'apprentissage automatique, traitement de données à grande échelle, analyse prédictive, etc.), qui vise à simplifier la gestion et le déploiement des packages. La distribution comprend des packages de science des données adaptés à Windows, Linux et macOS.