

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
SAAD DAHLEB BLIDA 01 UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE



MASTER'S INTELLIGENT SYSTEMS ENGINEERING

DATA MINING

PROJECT REPORT

Using the RANDOM FOREST DATA
MINING ALGORITHM on the
PhiUSIIL_Phishing_URL_Dataset

Made by :

ABDELATIF MEKRI
HALIMA NFIDSA
NAHLA YASMINE MIHOUBI

Academic year : 2023-2024

Contents

I.	ALGORITHME À IMPLÉMENTER	3
II.	PRINCIPE DE L'ALGORITHME.....	3
III.	AVANTAGES DE L'ALGORITHME.....	3
IV.	DOMAINES D'APPLICATIONS.....	4
V.	LE DATASET	4
VI.	OBJECTIF.....	4
VII.	IMPLÉMENTATION DE L'ALGORITHME	4
1.	CONFIGURATION DE L'ENVIRONNEMENT	4
2.	LES DONNÉES	6
2.1.	LECTURE DES DONNÉES	6
2.2.	ANALYSE DES DONNÉES.....	6
2.3.	ANALYSE UNIVARIÉE	6
2.4.	PRÉTRAITEMENT DES DONNÉES	8
2.4.1.	SUPPRESSION DES COLONNES NON PERTINENTES :.....	8
2.4.2.	TRANSFORMATION DE VARIABLE TEXTUEL EN NUMÉRIQUE:	8
2.4.3.	VALEURS FORTEMENT CORRÉLÉES	8
2.4.4.	LE TEST DU CHI CARRÉ	10
2.4.5.	SUPPRESSION DES VALEURS FORTEMENT CORRÉLÉES.....	10
3.	IMPLÉMENTATION DU MODÈLE	11
3.1.	DATA SPLIT	11
3.2.	LANCEMENT DU MODÈLE	12
3.3.	VISUALISATION DU PREMIERE ARBRE.....	12
4.	RÉSULTATS ET ÉVALUATION	13
4.1.	INTERPRÉTATION DES RÉSULTATS	13
4.1.2.	RAPPORT DU CLASSIFICATION	13
4.1.3.	MATRICE DE CONFUSION.....	14
4.1.3.	FEATURES IMPORTANCES	15
4.1.4.	COURBE ROC (Receiver Operating Characteristic).....	15
4.2.	COMPARAISON AVEC D'AUTRES MODÈLE	16
4.3.	INTERPRÉTATION DES RÉSULTATS.....	18
VIII.	CONCLUSION :	19

I. ALGORITHME À IMPLÉMENTER

Random Forest est un algorithme d'apprentissage supervisé polyvalent, utilisé pour la classification et la régression.

Il appartient à la famille des méthodes d'ensemble, qui combinent les prédictions de plusieurs modèles de base pour améliorer la performance prédictive.

Random Forest construit un ensemble de modèles d'arbres de décision lors de l'entraînement et effectue des prédictions en agrégeant les prédictions de ces arbres individuels.

II. PRINCIPE DE L'ALGORITHME

Random Forest combine les concepts de bagging (échantillonnage avec remplacement) et de sélection aléatoire des caractéristiques.

Pendant l'entraînement, il crée un grand nombre d'arbres de décision, chaque arbre étant formé sur un sous-ensemble aléatoire des données d'entraînement et des caractéristiques.

Lors de la prédiction, chaque arbre vote pour la classe majoritaire et la classe ayant le plus de votes est choisie comme prédiction finale.

Cette approche réduit le surajustement et améliore la robustesse du modèle.

Il est capable de gérer efficacement les deux types de problèmes. Dans les tâches de classification, il prédit les étiquettes de classe des instances, tandis que dans les tâches de régression, il prédit des valeurs continues.

Random Forest fait partie de la famille des méthodes d'ensemble, qui sont des techniques qui créent plusieurs modèles et combinent ensuite leurs prédictions pour produire de meilleurs résultats que n'importe quel modèle individuel

Pendant la phase d'entraînement, Random Forest construit un ensemble d'arbres de décision. Chaque arbre de décision est entraîné de manière indépendante sur un échantillon de l'ensemble de données d'origine.

Une fois que l'ensemble d'arbres de décision est entraîné, les prédictions sont effectuées en agrégeant les prédictions des arbres individuels. Dans les tâches de classification, chaque arbre "vote" pour la classe la plus populaire, et la classe avec le plus de votes devient la prédiction finale. Dans les tâches de régression, les prédictions des arbres individuels sont moyennées pour obtenir la prédiction finale.

III. AVANTAGES DE L'ALGORITHME

Random Forest est robuste au surajustement, grâce à l'effet de moyennage des multiples arbres.

Il gère bien les données de grande dimensionnalité et est moins sensible à la malédiction de la dimensionnalité.

Random Forest gère automatiquement les valeurs manquantes et maintient une précision même lorsque qu'une grande proportion des données est manquante.

Il fournit des estimations de l'importance des caractéristiques, permettant la sélection de caractéristiques et l'interprétation des résultats.

IV. DOMAINES D'APPLICATIONS

Détection de spam d'e-mails : identifier les e-mails frauduleux en fonction de diverses caractéristiques telles que les mots clés, les adresses IP, etc.

Prédiction de la santé des patients : prédire les maladies ou les diagnostics en fonction des données médicales et démographiques.

Classification de documents : classer automatiquement les documents en catégories basées sur leur contenu.

Systèmes de recommandation : recommander des produits ou du contenu en fonction du comportement passé des utilisateurs.

Analyse de la clientèle : segmenter les clients en groupes homogènes pour cibler les campagnes marketing, etc.

V. LE DATASET

Dataset utilisé : PhiUSIIL Phishing URL Dataset

Source : [<https://archive.ics.uci.edu/dataset/967/phiusiil+phishing+url+dataset>]

Description succincte du dataset :

PhiUSIIL Phishing URL est un ensemble de données conséquent comprenant 134 850 URL légitimes et 100 945 URL de phishing. La plupart des URL que nous avons analysées lors de la construction de l'ensemble de données sont parmi les plus récentes. Les caractéristiques sont extraites du code source de la page web et de l'URL.

Dataset statistics

Number of variables	56
Number of observations	74626
Missing cells	16
Missing cells (%)	< 0.1%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	31.9 MiB
Average record size in memory	448.0 B

Variable types

Text	5
Numeric	30
Categorical	21

VI. OBJECTIF

L'objectif de cette analyse est de prédire la classe de phishing pour les URLs en utilisant l'algorithme Random Forest.

La variable cible est la classe de phishing (1 pour phishing, 0 pour non-phishing).

VII. IMPLÉMENTATION DE L'ALGORITHME

1. CONFIGURATION DE L'ENVIRONNEMENT

Dans cette étape, nous avons installé et importé les différentes bibliothèques et packages nécessaires pour l'implémentation de notre algorithme.

Installing packages

```
[ ] pip install scikit-learn
    pip install pandas
    pip install numpy
    pip install matplotlib
    pip install seaborn
```

Importing Packages

```
[ ] from sklearn.ensemble import RandomForestClassifier
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import accuracy_score, classification_report, confusion_matrix ,roc_curve, auc
    from sklearn.tree import plot_tree
    from sklearn.datasets import load_iris
    import pandas as pd
    import numpy as np
    from ydata_profiling import ProfileReport
    import matplotlib.pyplot as plt
    import seaborn as sns
```

Package à importer	Utilité
<code>from sklearn.ensemble import RandomForestClassifier</code>	Cette instruction permet d'importer la classe RandomForestClassifier depuis le module sklearn.ensemble. Cette classe est utilisée pour entraîner des modèles de forêts aléatoires, pour les tâches de classification et de régression.
<code>from sklearn.model_selection import train_test_split</code>	La fonction train_test_split depuis le module sklearn.model_selection est essentielle pour diviser les ensembles de données en ensembles d'entraînement et de test, ce qui est crucial pour évaluer les performances des modèles .
<code>from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_curve, auc</code>	Pour importer plusieurs métriques et outils pour évaluer les performances des modèles d'apprentissage. Ces métriques comprennent la précision, le rapport de classification, la matrice de confusion, la ROC et l'aire sous la courbe ROC (AUC).
<code>from sklearn.tree import plot_tree</code>	Cette fonction est utilisée pour visualiser les arbres de décision .
<code>from sklearn.datasets import load_iris</code>	Cette fonction charge le célèbre jeu de données Iris, qui est souvent utilisé comme jeu de données jouet pour pratiquer les algorithmes d'apprentissage automatique.
<code>import pandas as pd</code>	La bibliothèque pandas est renommée en pd. Pandas est une bibliothèque puissante pour la manipulation et l'analyse de données ,surtout des données tabulaires.
<code>import numpy as np</code>	Numpy est une bibliothèque fondamentale pour le calcul numérique en Python, et elle fournit un support pour les tableaux multidimensionnels et les fonctions mathématiques.
<code>from ydata_profiling import ProfileReport</code>	Cette instruction d'importation importe la classe ProfileReport .Cette classe est utilisée pour générer des rapports d'analyse exploratoire de données pour les DataFrames pandas.
<code>import matplotlib.pyplot as plt</code>	La bibliothèque matplotlib est populaire en Python, et le module pyplot fournit une interface similaire à MATLAB pour créer des tracés et des visualisations.
<code>import seaborn as sns</code>	Seaborn est une bibliothèque de visualisation de données statistiques construite sur matplotlib, et elle fournit une interface de haut niveau pour créer des graphiques statistiques attrayants et informatifs.

2. LES DONNÉES

2.1. LECTURE DES DONNÉES

La lecture des données se fait à l'aide de la bibliothèque 'pandas' en important la méthode '.read_csv()' et en lui attribuant le chemin du jeu de données.

Reading csv

```
[ ] df= pd.read_csv("PhiUSIIL_Phishing_URL_Dataset.csv")  
  
df.head()
```

La méthode df.head() permet d'afficher les premières lignes de notre DataFrame .

2.2. ANALYSE DES DONNÉES

L'analyse des données joue un rôle central dans le processus du datamining. Elle consiste à explorer, à interpréter et à extraire des informations significatives à partir des ensembles de données disponibles. On utilise des techniques telles que l'analyse univariée , la visualisation de données et l'exploration de données pour identifier des tendances, des modèles ou des anomalies potentielles.

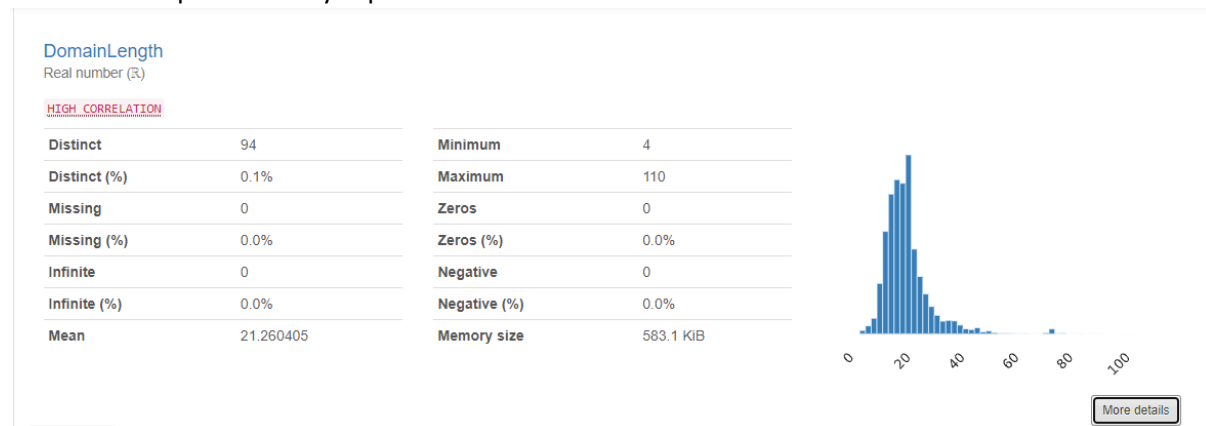
2.3. ANALYSE UNIVARIÉE

Grâce à la fonction 'profile', nous pouvons effectuer une analyse exploratoire qui vise à fournir une vue d'ensemble complète et détaillée des caractéristiques d'un ensemble de données . Nous avons choisi de la visualiser sous forme de page HTML.

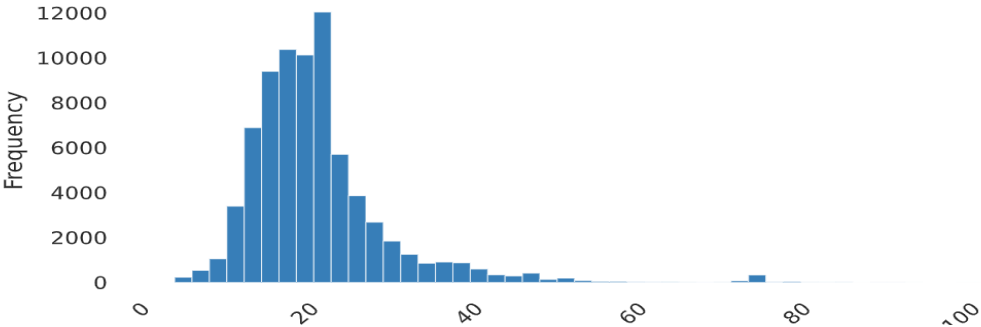
```
# Generate profile report  
profile = ProfileReport(df, title="Pandas Profiling Report", explorative=True)  
profile.to_file(output_file="REPORT.html")  
#Display the report as widgets  
#profile.to_widgets()  
  
print('Profiling Report found in "Pandas Profiling Report.html" ')
```

➡ Profiling Report found in "Pandas Profiling Report.html"

Voici un exemple de l'analyse possible avec la fonction :



Statistics	Histogram	Common values	Extreme values
Quantile statistics		Descriptive statistics	
Minimum	4	Standard deviation	8.8008333
5-th percentile	12	Coefficient of variation (CV)	0.41395417
Q1	16	Kurtosis	11.10282
median	20	Mean	21.260405
Q3	24	Median Absolute Deviation (MAD)	4
95-th percentile	37	Skewness	2.516038
Maximum	110	Sum	1586579
Range	106	Variance	77.454667
Interquartile range (IQR)	8	Monotonicity	Not monotonic

Statistics	Histogram	Common values	Extreme values
			
Statistics	Histogram	Common values	Extreme values
Value		Count	Frequency (%)
18		5414	7.3%
19		5110	6.8%
20		5013	6.7%
17		4953	6.6%
16		4818	6.5%
15		4578	6.1%
21		4366	5.9%
22		4139	5.5%
14		3797	5.1%
23		3530	4.7%
Other values (84)		28908	38.7%
Statistics	Histogram	Common values	Extreme values

Minimum 10 values

Maximum 10 values

Value	Count	Frequency (%)
4	80	0.1%
5	68	0.1%
6	78	0.1%
7	424	0.6%
8	110	0.1%
9	275	0.4%
10	771	1.0%
11	1448	1.9%
12	1941	2.6%

2.4. PRÉTRAITEMENT DES DONNÉES

2.4.1. SUPPRESSION DES COLONNES NON PERTINENTES :

Les colonnes 'FILENAME', 'URL', 'Domain' et 'Title' sont supprimées des données. Ces colonnes ne sont sous formes textuel , ce qui est pas evident pour l’algorithme Forets aleatoire , qui donne ces bonnes resultats que avec une forme numerique.

Ont a pouver les transformer sous forme numerique , mais le cout de transformation avec la technique de one-hot encoding avec ‘pd.get_dummies()’ ; convertit les variables catégorielles en vecteurs binaires, où chaque catégorie est représentée par un attribut binaire.

```
[ ] # Get the list of categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns.tolist()

# Print the list of categorical columns
print("Categorical Columns:")
for col in categorical_cols:
    print(col)

Categorical Columns:
FILENAME
URL
Domain
TLD
Title

[ ] df= df.drop(columns=['FILENAME', 'URL', 'Domain', 'Title'])
```

2.4.2. TRANSFORMATION DE VARIABLE TEXTUEL EN NUMÉRIQUE:

Pour la colonne "TLD", la transformation des données du format textuel au numérique se fait avec ‘pd.factorize’.

```
# Enumerate the values in the "TLD" column
df['TLD'], tld_enum = pd.factorize(df['TLD'])

# Print the enumerated values
print("Enumerated values for column 'TLD':")
print(df['TLD'])

# Print the unique values corresponding to the enumerated values
print("\nUnique values corresponding to the enumerated values:")
print(tld_enum)
```

2.4.3. VALEURS FORTEMENT CORRÉLÉES

La suppression des valeurs fortement corrélées car elles peuvent introduire des problèmes lors de l'analyse ou de la modélisation. Lorsque deux variables sont fortement corrélées, cela signifie qu'elles sont très similaires et qu'elles fournissent essentiellement la même information. En conséquence, conserver les deux variables pourrait entraîner une redondance dans le modèle et une utilisation inefficace des ressources computationnelles.

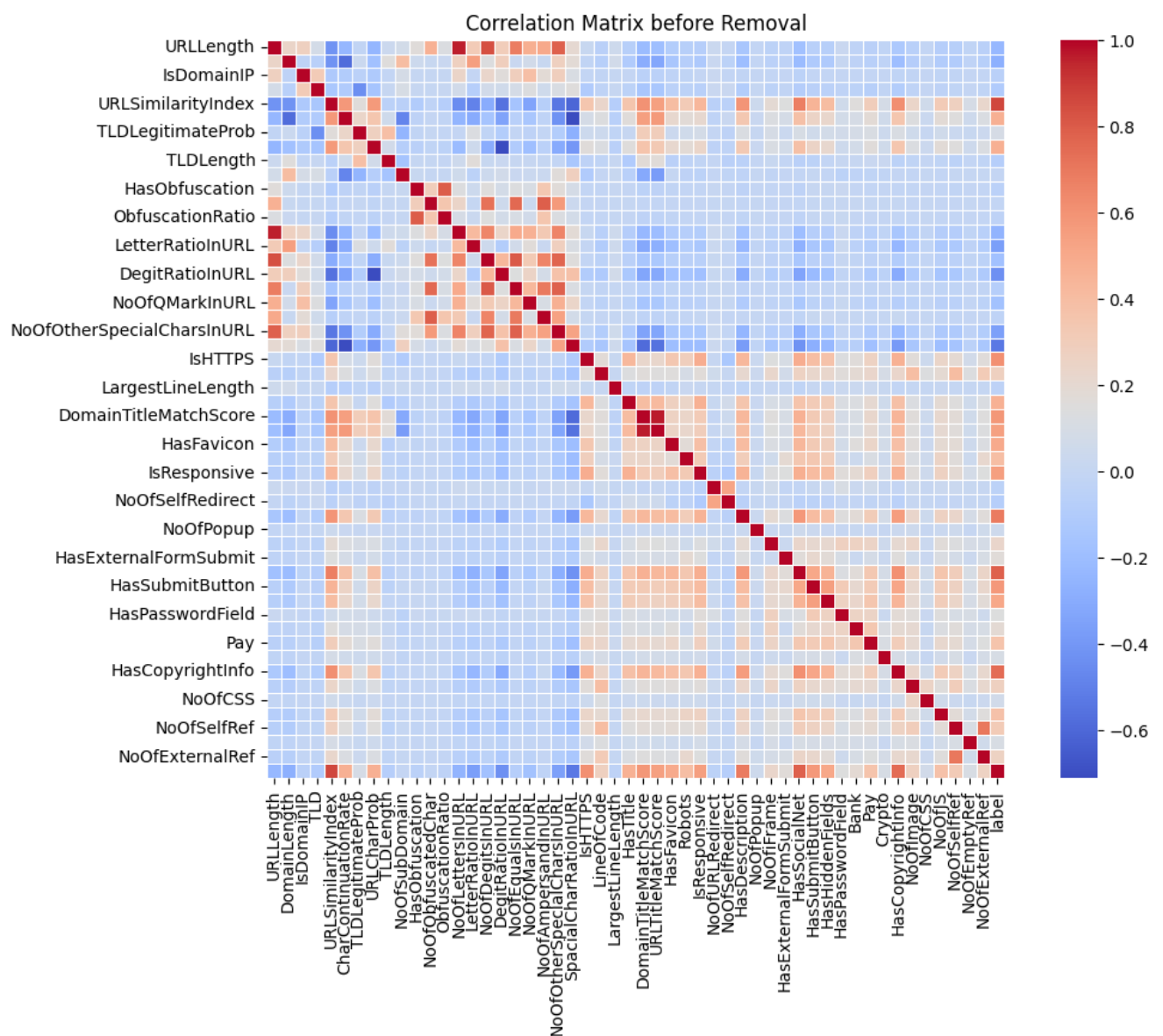
Pour détecter les valeurs fortement corrélées dans nos données, nous avons utilisé la fonction df.corr('pearson'). Cela nous a permis d'analyser la corrélation de Pearson entre toutes les paires de

variables numériques dans notre ensemble de données. En examinant la matrice de corrélation, nous avons identifié les paires de variables ayant des valeurs proches de 1 ou -1, ce qui indique une forte corrélation positive ou négative, respectivement .

```
# Calculate the correlation matrix before removing highly correlated features
correlation_matrix_before = df.corr('pearson')

# Plot the correlation matrix after removal
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix_before, annot=False, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Matrix before Removal')
plt.savefig('Correlation Matrix before Removal.png')
#show in png format
plt.imshow(plt.imread('Correlation Matrix before Removal.png'))
#plt.axis('off') # Optionally turn off axis
plt.show()
```

La matrice de corrélation avant la suppression des variables fortement corrélées :



2.4.4. LE TEST DU CHI CARRÉ

Le test du chi carré est un outil statistique pour déterminer si deux variables catégorielles sont liées d'une manière significative. En d'autres termes, il nous dit si la répartition observée des données est différente de ce à quoi on s'attendrait si les deux variables étaient indépendantes l'une de l'autre.

```
# Iterate over each column in the DataFrame
for col in df.columns:
    contingency_table = pd.crosstab(df[col], df['label'])
    chi2, p_value, _, _ = chi2_contingency(contingency_table)
    print(f"Chi-square test for {col}:")
    print(f"Chi-square statistic: {chi2}")
    print(f"P-value: {p_value}")
    print()
```

Toutes ces méthodes utilisées pour identifier les colonnes à supprimer.

2.4.5. SUPPRESSION DES VALEURS FORTEMENT CORRÉLÉES

En analysant les résultats de corrélation précédents, nous avons observé que la plupart des valeurs présentaient une faible corrélation, et suite à une recherche nous avons constaté que les meilleures valeurs se situent entre moins de 0.9 et 0.7 de corrélation. Nous avons choisi 0.7 comme seuil pour supprimer un maximum de variables afin d'améliorer les résultats lors de la mise en œuvre du modèle et de test .

```
# Set a threshold for correlation
threshold = 0.7 # value optimal between 0.7 & 0.9

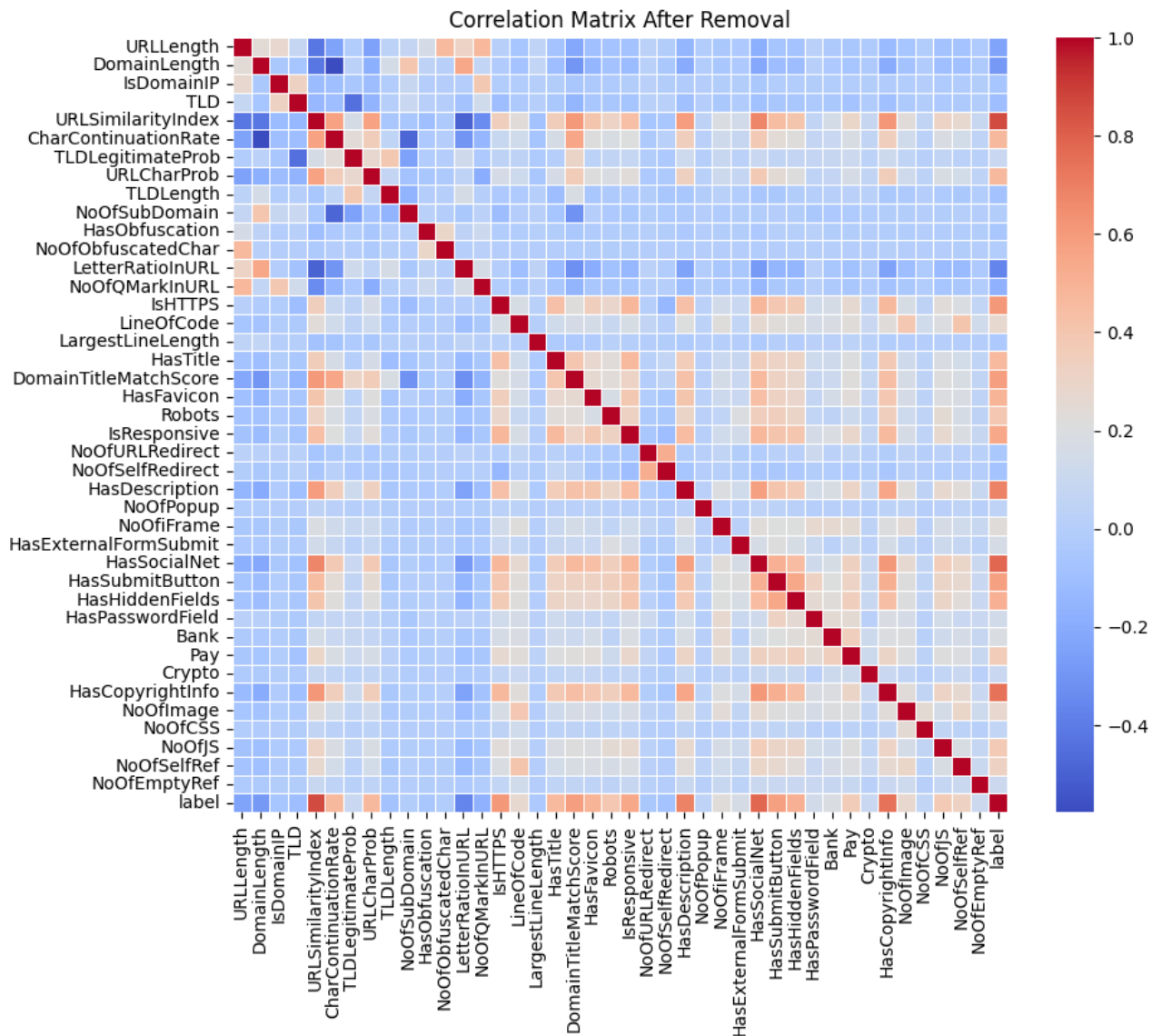
print("Removed features:")

# Find pairs of features with correlation greater than the threshold
correlated_features = set()
for _ in range(4):
    correlation_matrix = df.corr('pearson').abs()
    for i in range(len(correlation_matrix.columns)):
        for j in range(i):
            if i != j and abs(correlation_matrix.iloc[i, j]) > threshold:
                colname = correlation_matrix.columns[i]
                if colname != "label":
                    correlated_features.add(colname)
                    print(colname)
# Remove highly correlated features
# Check if the columns exist in the dataframe before dropping them
existing_columns = [col for col in correlated_features if col in df.columns]
df = df.drop(columns=existing_columns)
```

En utilisant la méthode drop de pandas, nous avons visé à supprimer les valeurs suivantes :

ObfuscationRatio	NoOfAmpersandInURL
NoOfLettersInURL	NoOfOtherSpecialCharsInURL
NoOfDegitsInURL	NoOfOtherSpecialCharsInURL
NoOfDegitsInURL	NoOfOtherSpecialCharsInURL
DegitRatioInURL	SpacialCharRatioInURL
NoOfEqualsInURL	URLTitleMatchScore
NoOfEqualsInURL	NoOfExternalRef

La matrice de corrélation après la suppression des variables fortement corrélées :



3. IMPLÉMENTATION DU MODÈLE

3.1. DATA SPLIT

Nous avons effectué `X = dfd.drop(columns=['label'])` pour créer une nouvelle variable X qui contient toutes les colonnes de notre DataFrame dfd, à l'exception de la colonne 'label'. Cette opération permet de séparer les caractéristiques (ou variables indépendantes) de la cible (ou variable dépendante).

Ensuite, nous avons effectué `y = dfd['label']` pour créer une nouvelle variable y qui contient uniquement la colonne 'label' de notre DataFrame dfd. Cette variable représente notre cible ,que nous cherchons à prédire à l'aide de nos caractéristiques.

```
3] X = dfd.drop(columns=['label'])
    y = dfd['label']
```

Division en Ensembles d'Entraînement et de Test :

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

X_train : Cet ensemble contient les caractéristiques pour l'entraînement du modèle.

X_test : Cet ensemble contient les caractéristiques pour tester le modèle.

y_train : Cet ensemble contient les valeurs correspondant à l'ensemble d'entraînement.

y_test : Cet ensemble contient les valeurs correspondant à l'ensemble de test.

‘**train_test_split(X, y, test_size=0.2, random_state=42)**’ divise les données en deux ensembles, en utilisant 80% des données pour l'entraînement (X_train, y_train) et 20% pour le test (X_test, y_test). Le paramètre **random_state=42** assure que la séparation des données est reproductible.

3.2. LANCEMENT DU MODÈLE

```
# Creating and training the Random Forest classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)

# Making predictions on the testing set
y_pred = rf_classifier.predict(X_test)

# Calculating the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Le code entraîne un classificateur de forêt aléatoire pour faire des prédictions et évaluer sa précision en calculant la précision du modèle.

- Crée une instance du classificateur de forêt aléatoire avec 100 arbres dans la forêt et une graine aléatoire fixée à 42 pour assurer la reproductibilité.
- entraîne le classificateur de forêt aléatoire sur les données d'entraînement. Le modèle apprend à partir des caractéristiques (X_train) et de leurs valeurs cibles correspondantes (y_train).
- ‘**rf_classifier.predict(X_test)**’ : Cette ligne utilise le modèle entraîné pour faire des prédictions sur les données de test (X_test). Les prédictions sont stockées dans y_pred.
- ‘**accuracy = accuracy_score(y_test, y_pred)**’ : Cette ligne calcule la précision du modèle en comparant les valeurs prédites (y_pred) aux valeurs cibles réelles (y_test) .
- print("Accuracy:", accuracy)**: Cette ligne affiche la précision du modèle sur les données de test. Plus la précision est proche de 1, plus le modèle est performant.

3.3. VISUALISATION DU PREMIERE ARBRE

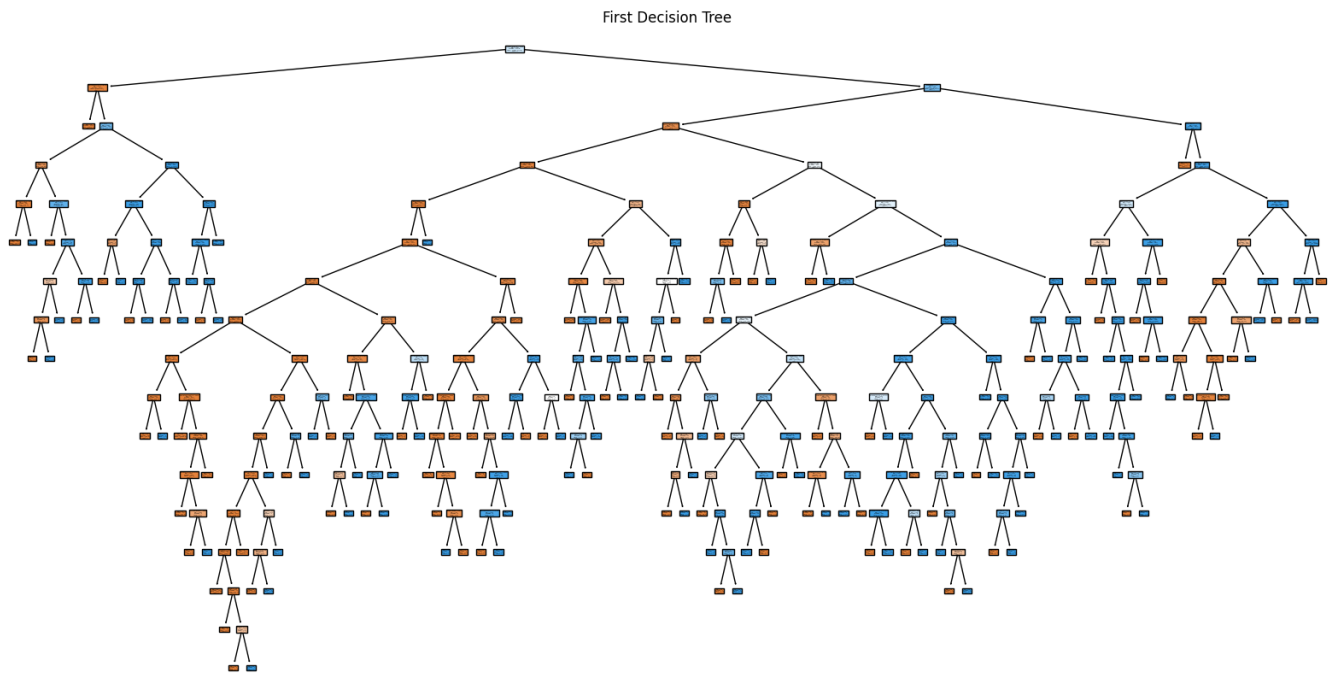
Pour la visualisation d’un arbre de décision à l'aide de la bibliothèque **matplotlib**, nous avons utiliser

la fonction 'plot_tree' du module 'tree' de scikit-learn. Voici comment vous pouvez le faire

```
# Get feature names from the DataFrame columns
feature_names = df.columns.tolist() # Assuming df is your DataFrame

# Plotting the first tree
plt.figure(figsize=(20,10))
plot_tree(tree, filled=True, feature_names=feature_names, class_names=class_names)
plt.title('First Decision Tree')
plt.show()
```

Nous avons obtenu la résultat suivante



4. RÉSULTATS ET ÉVALUATION

4.1. INTERPRÉTATION DES RÉSULTATS

4.1.2. RAPPORT DU CLASSIFICATION

```
# Generate a classification report
print("Classification Report:")
print(classification_report(y_test, y_pred,digits=20))
```

30]

.. Classification Report:

	precision	recall	f1-score	support
0	1.0000000000000000	1.0000000000000000	1.0000000000000000	20124
1	1.0000000000000000	1.0000000000000000	1.0000000000000000	27035
accuracy		1.0000000000000000	47159	
macro avg	1.0000000000000000	1.0000000000000000	1.0000000000000000	47159
weighted avg	1.0000000000000000	1.0000000000000000	1.0000000000000000	47159

Ensuite, on imprime le rapport de classification, qui fournit des détails sur les performances du modèle pour chaque classe, y compris la précision, le rappel, le score F1 et le support.

-La précision est le nombre de prédictions correctes parmi toutes les prédictions positives.

-Le rappel est le nombre de prédictions correctes parmi toutes les valeurs réelles positives.

-Le score F1 est une mesure de la précision et du rappel.

-Le support est le nombre d'occurrences de chaque classe dans l'ensemble de données de test.

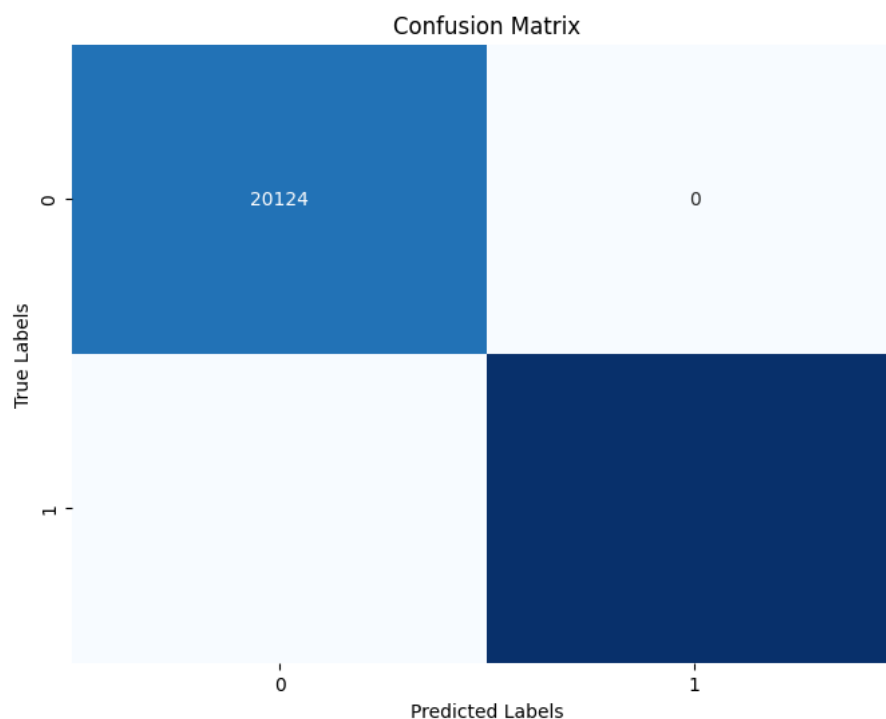
+ L'INTERPRÉTATION

La précision, le rappel et le score F1 sont tous de 1, ce qui indique une performance parfaite du modèle. L'exactitude globale du modèle est également de 1, ce qui confirme sa haute performance.

4.1.3. MATRICE DE CONFUSION

```
# Plotting the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, cmap='Blues', fmt='d', cbar=False)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
```

On trace une matrice de confusion pour évaluer les performances du modèle de classification. La matrice de confusion est une table qui montre les prédictions du modèle par rapport aux valeurs réelles de l'ensemble de test.



Les valeurs sur la diagonale principale représentent le nombre de prédictions correctes pour chaque classe. Par exemple, si la valeur à l'intersection de la ligne 0 et de la colonne 0 est 20124, cela signifie qu'il y a 20124 prédictions correctes pour la classe 0.

Les valeurs en dehors de la diagonale principale représentent les erreurs de classification. Par exemple, si la valeur à l'intersection de la ligne 0 et de la colonne 1 est 0, cela signifie qu'il n'y a aucune prédiction incorrecte de la classe 0 comme classe 1.

4.1.3. FEATURES IMPORTANCES

```
# Plotting feature importances
plt.figure(figsize=(10, 6))
feat_importances = pd.Series(rf_classifier.feature_importances_, index=X.columns)
feat_importances.nlargest(15).plot(kind='barh')
plt.title('Top 15 Most Important Features')
plt.xlabel('Relative Importance')
plt.ylabel('Features')
plt.show()
```

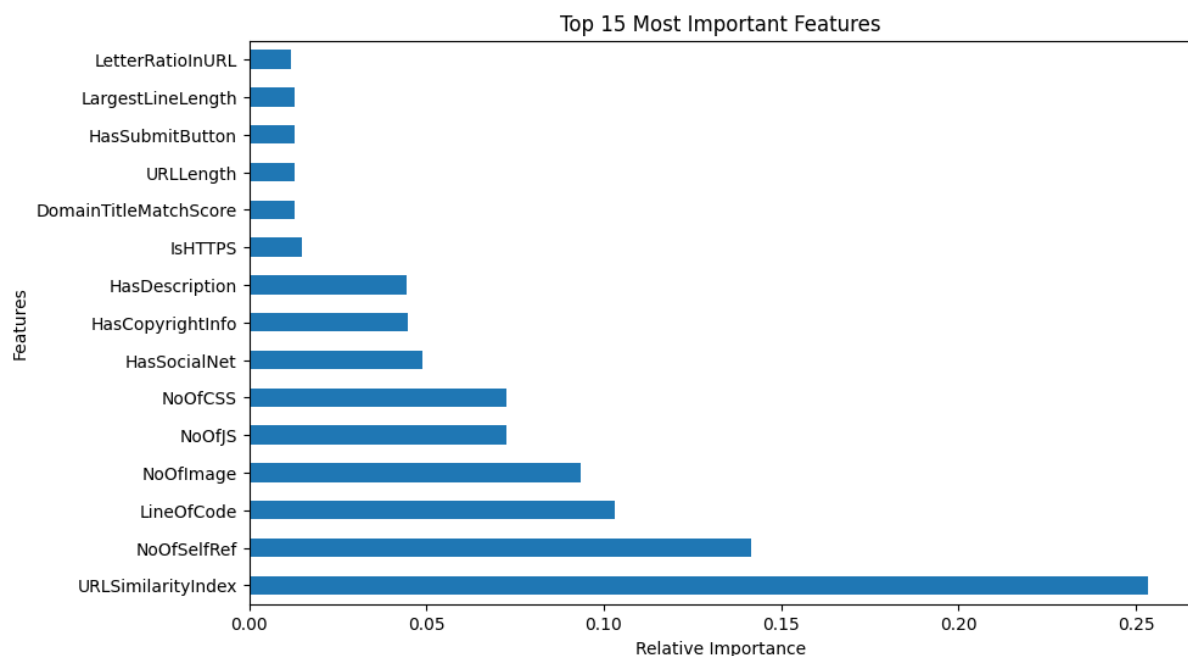
Le graphique des importances des caractéristiques pour le modèle de forêt aléatoire, nous permet de visualiser quelles caractéristiques ont le plus d'impact sur les prédictions du modèle. Voici une interprétation de ce graphique :

Chaque barre horizontale représente l'importance relative d'une caractéristique.

Les caractéristiques sont affichées sur l'axe y et leur importance relative est affichée sur l'axe x.

Les caractéristiques sont triées par importance, de la plus importante à la moins importante.

Exemple d'affichage des 15 caractéristiques les plus importantes



4.1.4. COURBE ROC (Receiver Operating Characteristic)

Cette visualisation nous permet d'évaluer la capacité du modèle à discriminer entre les classes positives et négatives en fonction de différents seuils de classification.

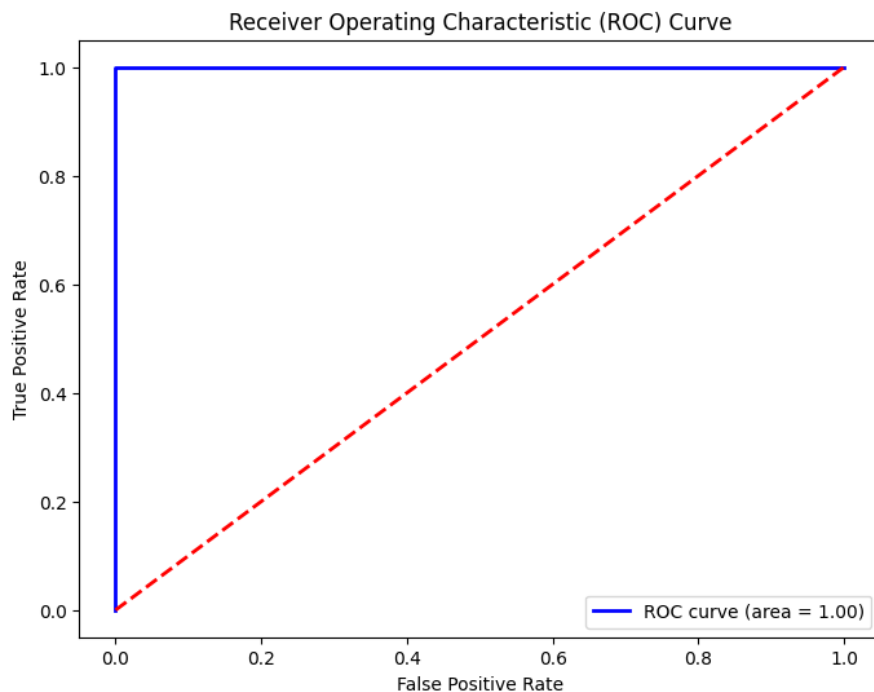
```
# Plotting ROC curve

probs = rf_classifier.predict_proba(X_test)
fpr, tpr, thresholds = roc_curve(y_test, probs[:,1])
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```

Une courbe ROC idéale se rapproche le plus possible du coin supérieur gauche du graphique, ce qui indique un taux élevé de vrais positifs et un faible taux de faux positifs.

L'aire sous la courbe ROC (AUC) mesure la capacité du modèle à distinguer les classes. Plus l'AUC est proche de 1, meilleures sont les performances du modèle.



4.2. COMPARAISON AVEC D'AUTRES MODÈLE

Nous avons implémenté les modèles suivants afin de tester la résilience du modèle de forêt aléatoire : les k plus proches voisins (KNN), les arbres de décision (DT), les machines à vecteurs de support (SVM) et la naïve bayésienne (NB)

<div>NB Naive Bayes</div>	<div>Naive Bayes Confusion Matrix</div> <table><tr><th></th><th>0</th><th>1</th></tr><tr><th>0</th><td>20118</td><td>6</td></tr><tr><th>1</th><td>0</td><td>1</td></tr></table>		0	1	0	20118	6	1	0	1
	0	1								
0	20118	6								
1	0	1								
<div>KNN K-Nearest Neighbors</div>	<div>KNN Confusion Matrix</div> <table><tr><th></th><th>0</th><th>1</th></tr><tr><th>0</th><td>20015</td><td>109</td></tr><tr><th>1</th><td>0</td><td>1</td></tr></table>		0	1	0	20015	109	1	0	1
	0	1								
0	20015	109								
1	0	1								
<div>DT Decision Tree</div>	<div>Decision Tree Confusion Matrix</div> <table><tr><th></th><th>0</th><th>1</th></tr><tr><th>0</th><td>20124</td><td>0</td></tr><tr><th>1</th><td>0</td><td>1</td></tr></table>		0	1	0	20124	0	1	0	1
	0	1								
0	20124	0								
1	0	1								
<div>SVM Support Vector Machine</div>	<div>SVM Confusion Matrix</div> <table><tr><th></th><th>0</th><th>1</th></tr><tr><th>0</th><td>20120</td><td>4</td></tr><tr><th>1</th><td>0</td><td>1</td></tr></table>		0	1	0	20120	4	1	0	1
	0	1								
0	20120	4								
1	0	1								

4.3. INTERPRÉTATION DES RÉSULTATS

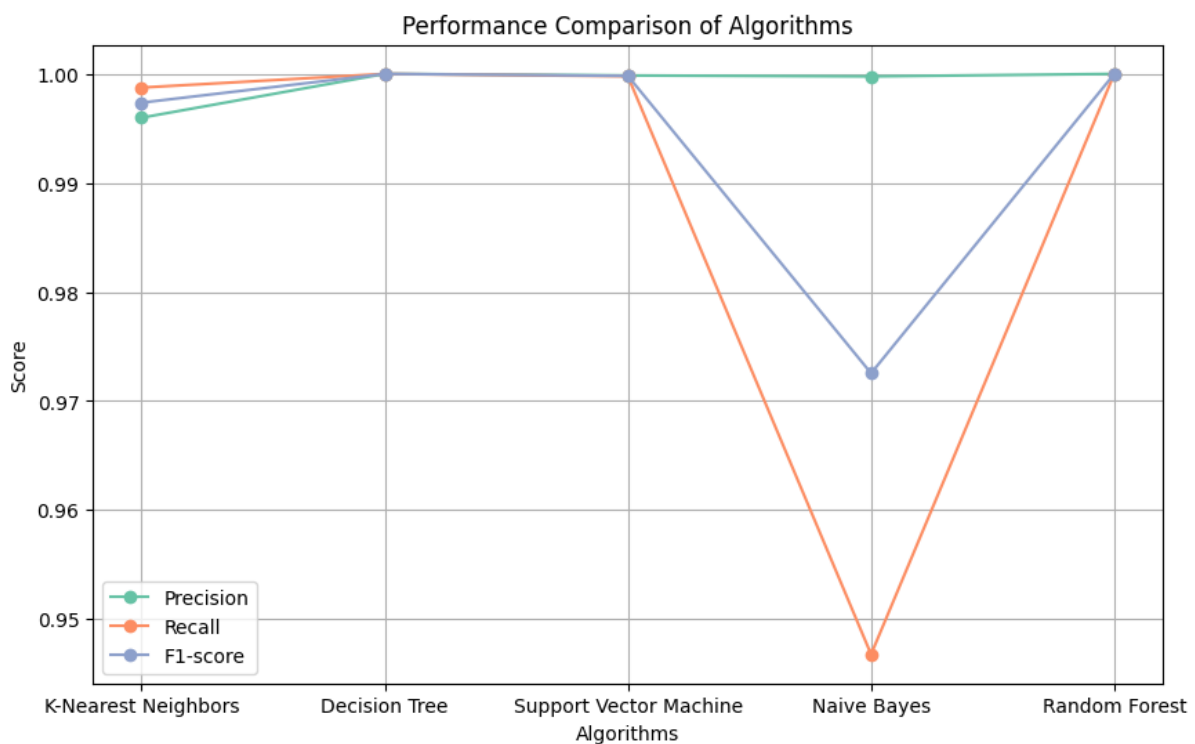
Comparons les résultats des modèles SVM, Naive Bayes, Arbre de Décision et k plus proches voisins (KNN) avec ceux du modèle de forêt aléatoire :

SVM : Précision de 99,98% et précision, rappel et score F1 de près de 1 pour les deux classes.

Naive Bayes : Précision de 96,94% et précision, rappel et score F1 de près de 0,97 pour les deux classes.

Arbre de Décision : Précision de 100% et précision, rappel et score F1 de 1 pour les deux classes.

KNN : Précision de 99,70% et précision, rappel et score F1 de près de 0,997 pour les deux classes.



Comparé à ces modèles, le modèle de forêt aléatoire a une performance similaire ou légèrement supérieure en termes de précision, de rappel et de score F1 pour les deux classes, avec une précision de 100% et des valeurs de précision, de rappel et de score F1 de 1 pour les deux classes. Cela indique que le modèle de forêt aléatoire est très performant dans cette tâche de classification, avec une précision parfaite pour les données de test.

VIII. CONCLUSION :

Dans ce projet, nous avons réalisé une analyse approfondie des données pour construire et évaluer plusieurs modèles de classification dans le but de prédire les URLs de phishing. Voici les points clés de notre conclusion :

Analyse des données : Nous avons exploré et prétraité le jeu de données, en extrayant des fonctionnalités pertinentes à partir du code source des pages web et des URL.

Construction des Modèles : Nous avons entraîné plusieurs modèles de classification, y compris la forêt aléatoire, le SVM, le Naive Bayes, l'arbre de décision et le k plus proches voisins (KNN), en utilisant les fonctionnalités extraites comme variables d'entrée.

Évaluation des Modèles : Nous avons évalué les performances des modèles en utilisant des mesures telles que la précision, le rappel, le score F1 et l'exactitude. Nous avons également visualisé les résultats à l'aide de graphiques tels que les courbes ROC et les matrices de confusion.

Comparaison des Modèles : Nous avons comparé les performances des différents modèles et avons constaté que la forêt aléatoire avait les performances les plus élevées, avec une précision de 100% et des scores F1 parfaits pour les deux classes.

Conclusions et Perspectives : En conclusion, le modèle de forêt aléatoire s'est avéré être le plus performant pour la tâche de prédiction des URLs de phishing dans notre ensemble de données. Cependant, il existe toujours des possibilités d'amélioration, telles que l'exploration de techniques avancées de prétraitement des données, l'optimisation des hyperparamètres des modèles et l'expansion de l'ensemble de données pour une meilleure généralisation.

En résumé, ce projet a démontré l'efficacité des techniques d'apprentissage automatique pour la détection des URLs de phishing, avec la forêt aléatoire se démarquant comme le modèle le plus performant.

