Projet de recherche (PRe)

Specialisation : mathématiques appliquées

Année universitaire : 2024/2025

Prévision de l'évolution future de la couverture végétale en Tunisie dans le contexte du changement climatique

---

Rapport non confidentiel

Auteur : MAALEJ Omar

Promotion : 2025

Superviseur à l'ENSTA Paris :
Zacharie Ales

Superviseur au laboratoire ESE :
Soudani Kamel, Pr
Université Paris Saclay

Période de stage du 03/06/2024 **au** 30/08/2024

Organisme d'accueil : Laboratoire Écologie, Systématique et Évolution à l'Université Paris-Saclay

Adresse : 12 rue 128, 91190 Gif-sur-Yvette, France

Note de non-confidentialité

Ce document ne contient aucune information confidentielle et peut être publié sur internet.

Résumé

Ce rapport de stage traite de la prévision de l'évolution future de la couverture végétale en Tunisie en utilisant des méthodes avancées de machine learning. L'étude se concentre sur l'analyse des données climatiques et environnementales pour comprendre l'impact du changement climatique sur les écosystèmes forestiers tunisiens. Le rapport détaille les processus d'extraction, de traitement et d'analyse des données, ainsi que l'utilisation de plusieurs modèles et algorithmes de machine learning, tels que Random Forest, XGBoost et SVM. Ces modèles sont entraînés sur les données disponibles et adaptés à notre problématique pour fournir des prédictions futures précises. Chaque modèle est évalué en termes de performances prédictives afin de sélectionner celui qui offre la meilleure précision et performance globale.

Abstract

This internship report focuses on predicting the future evolution of vegetation cover in Tunisia using advanced machine learning methods. The study concentrates on analyzing climatic and environmental data to understand the impact of climate change on Tunisian forest ecosystems. The report details the processes of data extraction, processing, and analysis, as well as the use of various machine learning models and algorithms, such as Random Forest, XGBoost, and SVM. These models are trained on the available data and tailored to our specific problem to provide accurate future predictions. Each model is evaluated based on its predictive performance to select the one that offers the best accuracy and overall performance.

Keywords

Forest Cover, Climate Change, Tunisia,Forecasting,
Machine Learning,Random Forest,XGBoost, SVM (Support Vector Machine), Réseaux de neurones

Remerciements

Je suis profondément reconnaissant pour l'expérience et les connaissances inestimables que j'ai acquises lors de mon stage au sein du laboratoire ESE. J'exprime ma sincère gratitude envers M. SOUDANI Kamel, Professeur des Universités, UFR des Sciences d'ORSAY (Université Paris Saclay), mon mentor, pour son accompagnement constant, son soutien et ses encouragements tout au long de ce parcours. Son expertise et ses conseils ont joué un rôle crucial dans la formation de ma compréhension du sujet. Mes sincères remerciements vont également à toute l'équipe de ESE pour avoir créé un environnement propice aux interactions collaboratives et à l'apprentissage. Les réunions hebdomadaires en laboratoire, où les collègues échangent sur leurs travaux, ont été précieuses pour approfondir ma compréhension des divers domaines de recherche. Les informations, idées et connaissances partagées lors de ces rencontres ont considérablement enrichi mon expérience de stage. En outre, je remercie l'organisation de m'avoir offert cette opportunité remarquable d'explorer le monde de la recherche scientifique. Je tiens à exprimer aussi mes sincères remerciements à M. Zacharie ALES pour étre mon enseignant referrant dont l'accompagnement et les conseils ont grandement contribué à la pertinence de mon stage. Ce stage a été une expérience transformative, et j'ai hâte d'appliquer les précieuses leçons et compétences acquises dans mes futures activités.

Table des matières

Table des figures

Introduction

La préservation des ressources forestières est devenue une priorité urgente à l'échelle mondiale, en réponse aux défis croissants posés par le changement climatique. Ce rapport présente les résultats d'un stage de recherche axé sur la prévision de l'évolution future de la couverture forestière en Tunisie à l'aide de méthodes de machine learning avancées. Au-delà de l'analyse des données climatiques et environnementales, cette étude vise également à sensibiliser sur l'importance critique du changement climatique, sa rapidité et son impact profond sur la nature. Les modèles prédictifs développés dans cette recherche visent à fournir des insights essentiels sur la manière dont les écosystèmes forestiers tunisiens pourraient répondre aux changements climatiques futurs, offrant ainsi une base solide pour la formulation de politiques de conservation adaptatives. En mettant en lumière ces enjeux, cette étude aspire à mobiliser une action concertée pour protéger et restaurer les forêts face aux pressions environnementales croissantes.

Chapitre 1

Extraction et analyse des donneés

1.1 Introduction

Dans ce chapitre, je vais aborder le processus d'extraction et d'acquisition des données, en décrivant en détail le travail effectué étape par étape.

1.2 Analyse génerale

Pour effectuer des prévisions précises, il est crucial de disposer de données climatiques solides. En général, la végétation est influencée principalement par la température, les précipitations, le rayonnement solaire et la richesse minérale du sol. Pour ce projet, nous avons identifié des données climatiques pertinentes pour l'Afrique à partir des données climatiques actuelles et simulations futures WorldClim [4].
Les données climatiques actuelles sont des moyennes couvrant la période de 1970 à 2000 et comprennent les moyennes des 19 variables climatiques sur cette periode, chacune représentée par une carte géographique qui couvre le globe avec des mesures prises tous les kilomètres carrés. La signification de chaque variable est explicitée ci-dessous :

```
BIO1: Température moyenne annuelle
BIO2 : Plage diurne moyenne (Moyenne mensuelle de la différence
entre la température maximale et minimale)
BIO3 : Isothermalité (BIO2/BIO7) (×100)
BIO4 : Saisonnalité de la température (écart type ×100)
BIO5 : Température maximale du mois le plus chaud
BIO6 : Température minimale du mois le plus froid
BIO7 : Plage annuelle de température (BIO5 - BIO6)
BIO8 : Température moyenne du trimestre le plus humide
BIO9 : Température moyenne du trimestre le plus sec
BIO10 : Température moyenne du trimestre le plus chaud
BIO11 : Température moyenne du trimestre le plus froid
BIO12 : Précipitations annuelles
BIO13 : Précipitations du mois le plus humide
BIO14 : Précipitations du mois le plus sec
BIO15 : Saisonnalité des précipitations (coefficient de variation)
BIO16 : Précipitations du trimestre le plus humide
```

```
BIO17 : Précipitations du trimestre le plus sec
BIO18 : Précipitations du trimestre le plus chaud
BIO19 : Précipitations du trimestre le plus froid
```

Pour la végétation, nous avons utilisé les cartes de végétation GLAD (Global Land Analysis and Discovery laboratory) de l'Université du Maryland, réalisée à partir d'images satellitaires Landsat [5]. Ces cartes sont fournies à 30 m de résolution spatiale (pixel). La classe de végétation est codée entre 0 et 19, dont la signification est explicitée ci-dessous.
Dans notre projet, nous avons choisi les deux cartes 40N-010E et 40N-000E, couvrant toute la Tunisie.

```
0: No data
1: True desert
2: Semi-arid
3: Dense short vegetation
4: Open tree cover
5: Dense tree cover
6: Tree cover gain
7: Tree cover loss, not fire
8: Salt pan
9: Wetland sparse vegetation
10: Wetland dense short vegetation
11: Wetland open tree cover
12: Wetland dense tree cover
13: Wetland tree cover gain
14: Wetland tree cover loss
15: Ice
16: Water
17: Cropland
18: Built-up
19: Ocean
```

Pour nos prévisions, nous avons utilisé les données bioclimatiques futures disponibles sur le site **WorldClim** [17] pour la période de 2021 à 2040, en se basant sur le scénario **SSP 245**. Ce scénario projette un avenir où les émissions de gaz à effet de serre augmentent de manière modérée, avec une mise en œuvre progressive mais partielle des politiques climatiques. Il anticipe une croissance démographique modérée, un développement économique stable avec des variations régionales, et bien que des efforts pour réduire les émissions soient présents, ils ne sont pas aussi ambitieux que dans d'autres scénarios SSP. Cela entraîne un réchauffement climatique continu mais moins rapide que dans les scénarios où les émissions sont plus élevées.

1.3 Analyse détaillé des variables

Dans un premier temps, nous avons extrait les cartes suivantes sur toute la Tunisie :
- 19 cartes climatiques, chacune représentant une variable spécifique, avec une résolution géographique de 30 secondes (1 km^2).
- Deux cartes géographiques contenant la classe de couverture végétale couvrant la Tunisie et une partie de l'Afrique (40N_010E et 40N_000E).
- Une carte de l'Afrique [7]..

L'extraction a été réalisée grâce au logiciel **QGIS**, un logiciel de système d'information géographique (SIG) open source, permettant de créer, visualiser, analyser et gérer des données géospatiales. En exploitant le manuel d'utilisation [11] pour comprendre les fonctionnalités disponibles dans ce logiciel, les opérations suivantes ont été réalisées :

- Extraction du contour géographique de la Tunisie à partir de la carte de l'Afrique. Notons que les données sont fournies initialement à l'échelle globale.

- Création de la grille : J'ai créé une grille de pixels à partir de cette carte, où chaque pixel représente 1 km², alignée avec les variables climatiques calculées pour chaque km². Cette grille a été générée à l'aide de l'outil **"Créer une grille à partir d'une couche"** de QGIS.

- Ajout de coordonnées : Dans la table attributaire de la grille, j'ai ajouté des colonnes pour les latitudes et longitudes de chaque pixel en utilisant l'outil de **"calcul d'expression"**.

- Fusion des couches de végétation : J'ai fusionné les deux couches de végétation en une seule, nommée "fusion_E_N", pour les intégrer ensuite à la carte de la Tunisie.

- Attribution de la classe de végétation : pour chaque pixel de la grille, nous avons rencontré le défi suivant : la grille et les couches climatiques avaient une résolution de 1 km², tandis que la classe de végétation est fournie tous les 30 mètres. Avec l'outil **"Statistiques zonales"** de QGIS, nous avons fusionné la couche grille (contenant les coordonnées des pixels) avec la couche de végétation fusionnée, en utilisant l'option **"majorité"** pour attribuer à chaque pixel de la grille la classe de végétation majoritaire de tous les pixels de 30 m² dans un pixel de 1 km². Ceci correspond à une mise à l'échelle de la carte de végétation de 30 m² à la résolution des données climatiques fournies à 1 km².

- Intégration des variables climatiques : Nous avons superposé les 19 couches de variables climatiques à la grille, en procédant couche par couche, et avons enregistré toutes les informations dans un fichier Excel.

Ainsi, nous avons obtenu le fichier Excel souhaité, contenant les coordonnées de chaque pixel, la classe de végétation majoritaire et les 19 variables climatiques associées.

Un fichier de même structure a été créé à partir des simulations fournissant les variables climatiques sous SSP 245. Ces deux fichiers ont servi en inputs aux codes python décrits dans les chapitres suivants.

Chapitre 2

Traitement des données

2.1 Introduction

Après avoir obtenu les données sous forme de fichier Excel, j'ai procède à leur analyse et leur traitement, notamment le nettoyage des données, l'analyse de la covariance des variables d'entrée afin de réduire leur nombre, ainsi que la normalisation et le traitement de la variable cible "Classe de végétation à prédire" par la fusion de certaines classes selon mes besoins en code Python, à l'aide de la bibliothèque prédéfinie sous Python : Pandas

2.2 Traitement des données d'entrée

2.2.1 Nettoyage des données

Lors de l'acquisition des données à partir de QGIS, certaines lignes du fichier Excel contenaient des valeurs nulles représentant des pixels correspondant à la mer Méditerranée, au sud de la Tunisie, ou à des pays voisins. Pour commencer, j'ai nettoyé les données en éliminant les lignes contenant des valeurs manquantes. Après le nettoyage, j'ai obtenu un tableau avec 349403 lignes, où chaque ligne représente un pixel de 1 km², et 22 colonnes : 2 colonnes représentent la latitude et la longitude du centre du pixel de 1 km², 19 colonnes représentent les 19 variables climatiques, et une colonne correspond à la classe de végétation GLAD2019. Notons que l'objectif est de modéliser empiriquement la relation existante entre la classe de végétation actuellement présente et les données climatiques actuelles. Une fois cette relation établie, elle est utilisée pour prédire la végétation future (en termes de présence/absence) en utilisant les simulations climatiques futures comme inputs de cette relation.

2.2.2 Réduction du dimensionnalité

Pour mieux comprendre notre jeu de données et sélectionner les meilleures variables, il est important d'examiner la dépendance entre les variables climatiques puisque la présence d'une forte correlation entre les variables peut entraîner une instabilité du modèle et des estimations de coefficients peu fiables. Par exemple, les modèles de régression linéaire supposent l'indépendance entre les variables d'entrée, et la présence de multicolinéarité peut rendre le modèle très sensible aux modifications

des données d'entrée, ce qui conduit généralement à de mauvaises performances sur les données de test.

En termes d'interprétabilité, la multicolinéarité complique la détermination de la contribution individuelle de chaque prédicteur aux prédictions du modèle, rendant difficile la compréhension de l'importance de chaque variable dans la prédiction [12].

Pour examiner les relations entre les variables climatiques et détecter les multicolinéarités, j'ai affiché la matrice de corrélation, qui montre la corrélation (coefficient de corrélation de Pearson) entre chaque couple de variables parmi les 19 variables climatiques. Cet outil statistique mesure la variance conjointe de plusieurs variables, aide à comprendre les relations entre elles, détecte les multicolinéarités et permet de réduire la dimensionnalité.

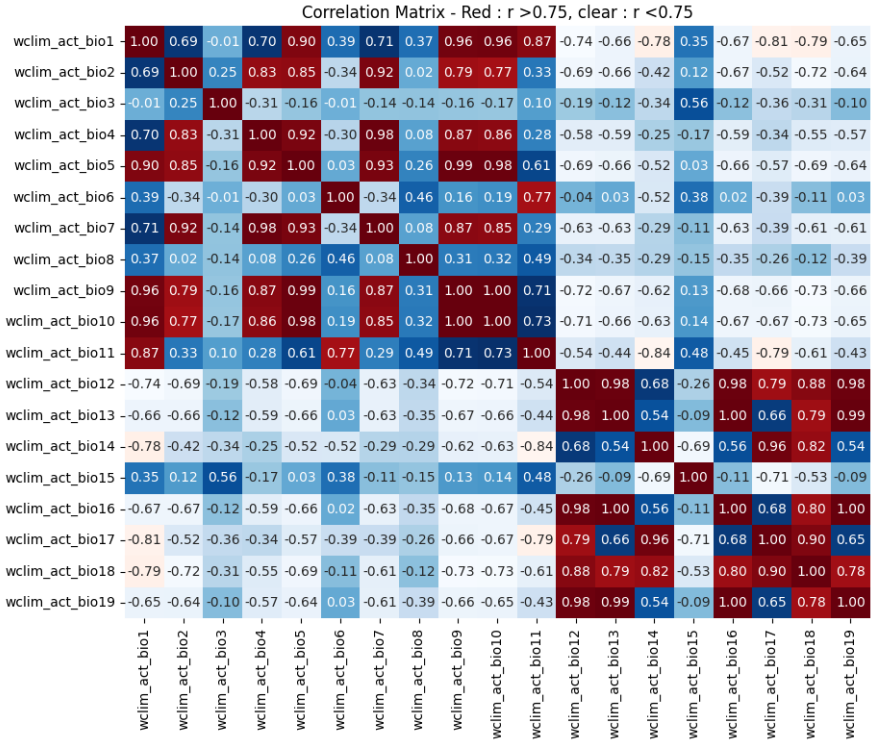

FIGURE 2.1 – Matrice de correlation

Comme l'indique la matrice de corrélation, il existe de fortes corrélations entre les variables de 1 à 11 et entre celles de 12 à 19. Cela est logique, car les 11 premières variables sont liées à la température, tandis que les variables de 12 à 19 concernent les précipitations. Il est donc naturel d'observer ces fortes corrélations au sein de chaque groupe.

Suite à l'analyse de cette matrice de corrélation, j'ai sélectionné les variables les moins corrélées entre elles, ainsi que celles qui sont les plus importantes du point de vue du climat tunisien. Ainsi, j'ai retenu les variables 3, 4, 6, 8, 11, 15, 17 et 19. De plus, j'ai ajouté les variables 2 et 12, qui représentent respectivement la température annuelle moyenne et les précipitations annuelles moyennes, car elles sont facilement interprétables et cruciales pour la végétation en Tunisie.

La nouvelle matrice de corrélation pour les variables sélectionnées est montrée ci-dessous :

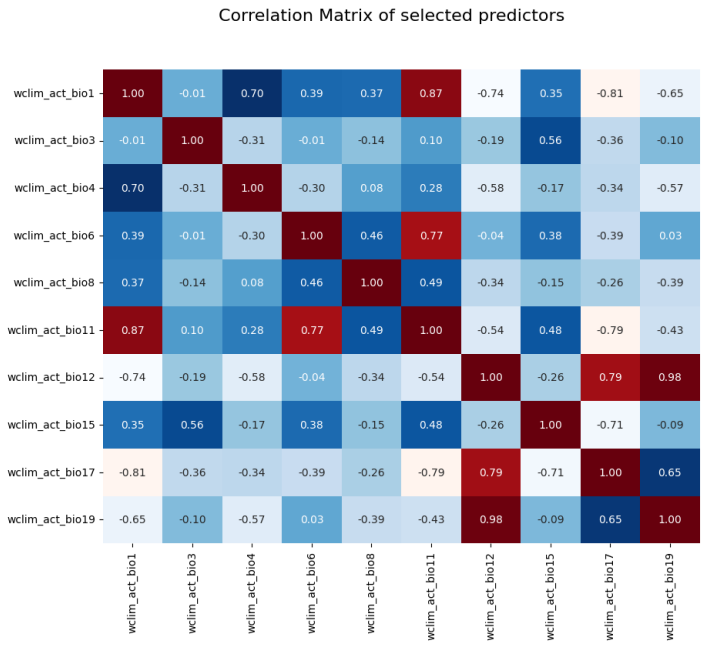

FIGURE 2.2 – Matrice de correlation pour les variables sélectionnées

Suite à cette démarche, j'ai créé un nouveau tableau comprenant uniquement les variables climatiques sélectionnées à l'aide de la matrice de corrélation, ainsi que la latitude, la longitude et la variable cible correspondant à la classe de végétation GLAD2019.

2.2.3 Standardisation

Les statistiques descriptives des variables sélectionnées sont illustrées ci-dessous :

	X	Y	BIO_1	BIO_3
mean	9.36201	33.448218	20.160296	39.391965
std	1.08353	1.982686	2.505628	1.926026
min	7.53750	30.245830	11.475000	27.117115
max	11.59583	37.537500	24.245832	46.175800

	BIO_4	BIO_6	BIO_8	BIO_11
mean	727.647350	4.798335	13.704778	11.277793
std	89.565574	1.403386	2.811277	1.697759
min	486.509491	-1.400000	4.183333	3.333333
max	885.754578	10.200000	23.083332	13.900000

	BIO_12	BIO_15	BIO_17	BIO_19
mean	204.754358	56.166239	13.113451	75.588341
std	204.693868	12.846725	16.953972	86.720579
min	27.000000	21.955446	0.000000	10.000000
max	1406.000000	87.420166	66.000000	605.000000

D'après ces résultats, on constate que certaines variables présentent des écarts d'échelle significatifs. Par exemple, la variable BIO1 varie entre 7 et 12, tandis que la variable BIO19 varie entre 10 et 605. Ce problème pose des défis, car certains modèles sont très sensibles à la mise à l'échelle des variables [16]. A titre d'exemple, les modèles basés sur la méthode de descente de gradient pour l'optimisation des poids (w_{ij}) comme la régression linéaire, la régression logistique, et les réseaux de neurones nécessitent que les données soient standardisées. En effet, en examinant la formule de la descente de gradient pour une fonction de coût typique comme celle de la régression linéaire, la mise à jour du poids w_j peut être écrite comme :

$w_j := w_j - \alpha \frac{\partial J}{\partial w_j}$

pour : $J = \mathbf{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - h_\theta(x_j^{(i)}))^2$: l'erreur quadratique moyenne, cela devient :

$w_j := w_j - \alpha \sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}$

où :

— m est le nombre d'exemples d'entraînement

— $h_\theta(x^{(i)})$ est la prédiction du modèle pour l'exemple i

— $y^{(i)}$ est la valeur réelle pour l'exemple i

— $x_j^{(i)}$ est la valeur de la j-ème caractéristique pour l'exemple i

On constate que $x_j^{(i)}$ est présent dans la formule, donc affecte la mise à jour des poids. La différence entre les plages de valeurs des caractéristiques entraînera des tailles de pas différentes pour chaque caractéristique, ce qui peut causer des problèmes de convergence. Pour garantir que la descente de gradient se déplace par petits pas vers le minimum, il faut mettre à l'échelle les données.

La normalisation des données n'affecte pas les modèles basés sur les arbres de décision. En effet, les arbres de décision construisent des séparations basées sur des seuils de caractéristiques et sont invariants par rapport aux transformations monotones des données.

Dans cette étude, la standardisation a été réalisée en utilisant la z-normalisation à l'aide de l'outil **"StandardScaler"** de la bibliothèque **scikit-learn**. Son principe consiste à prendre chaque variable x , soustraire sa moyenne μ , puis diviser par son écart type σ, comme indiqué par la formule : $z = \frac{x-\mu}{\sigma}$

Cette opération permet de centrer les données autour de zéro et de les mettre à l'échelle de manière à avoir une moyenne nulle et un écart type égal à un, facilitant ainsi l'analyse et l'entraînement des modèles statistiques et d'apprentissage machine.

2.3 Traitement de la variable réponse : classe de végétation GLAD2019

2.3.1 Analyse du besoin

Pour mieux comprendre la répartition de chaque classe de végétation en Tunisie à partir des données GLAD2019, qui comprennent des nombres entiers de 0 à 19 représentant chaque classe de végétation comme décrit dans le chapitre précédent, le nombre d'occurrences de chaque classe est montré ci-dessous :

```
GLAD2019
```

```
0     131690
1     125915
2      46070
17     26255
3      10469
18      2811
4       2564
8       2134
16       887
5        329
9        147
10        56
6         41
12        23
7          6
11         5
13         1
Name: count, dtype: int64
```

La classe majoritaire est 0 représente principalement des zones sans végétation (zones résidentielles, surfaces d'eau diverses, sols nus, etc.). Dans un premier temps, nous avons distingué quatre classes de végétation forestière mais en raison de la taille trop faible de certaines classes (4 et 5), une classe unique reproupant les classes 3, 4 et 5 a été créée. La carte GLAD a donc été simplifiée en une classification composée de trois classes :

- **Classe 0 :** regroupe les classes 0, 1, 2, 6, 7, 8, 9, 10, 11, 12, 13, 16, 18
- **Classe 1 :** comprend la classe 3 (végétation basse et dense), ainsi que les classes 4 et 5 (Forêts).
- **Classe 2 :** correspond à la classe 17, qui représente les zones de culture.

La nouvelle classification GLAD2019 est codée en 0, 1 et 2 correspondant aux classes décrites ci-dessus.

2.3.2 Problème de deséquilibre des données

Aprés avoir regroupé les 19 classes en 3 classes, la distribution statistique est montrée ci-dessous :

```
    GLAD2019
0     309786
2      26255
1      13362
Name: count, dtype: int64
```

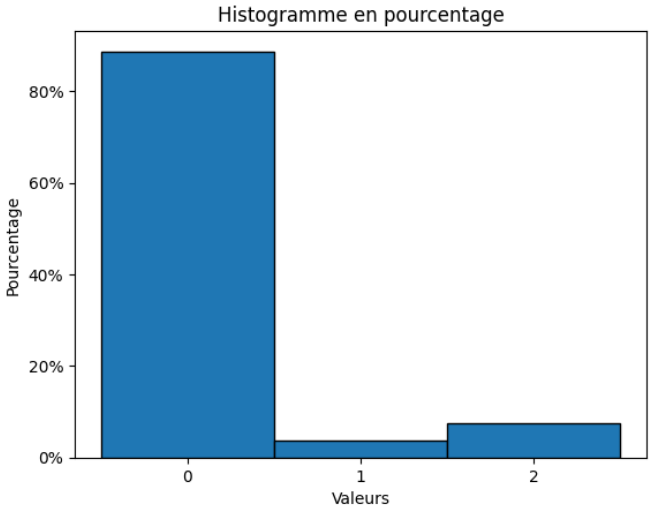

FIGURE 2.3 – Histogramme des classes

Le diagramme en camembert (Figure 2.3) représente la proportion de chaque classe. La répartition géographique est donnée dans la Figure 2.4.

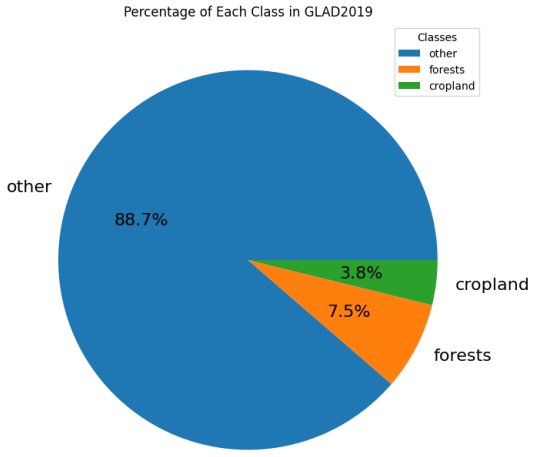

FIGURE 2.4 – Diagramme en camembert

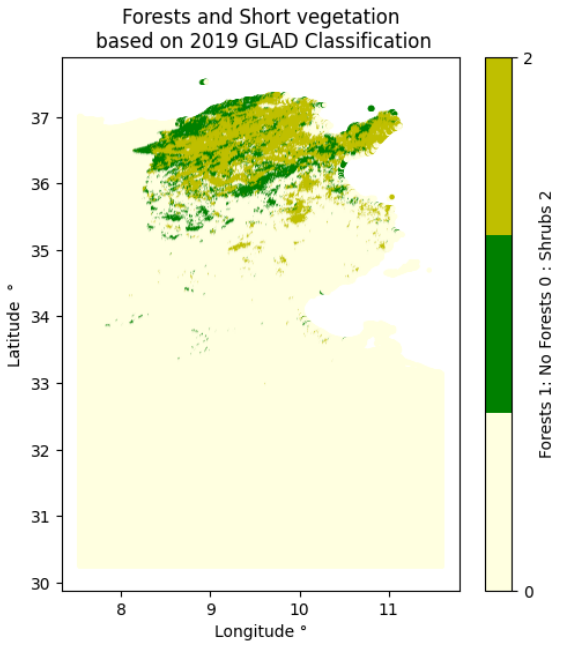

FIGURE 2.5 – Répartition simplifiée de la couverture végétale en Tunisie

La classe 0 est majoritaire et occupe 88,7 % de la surface de la Tunisie, tandis que les classes 1 et 2 occupent respectivement seulement 3,8 % et 7,5 %. La répartition géographique de la vegétation en Tunisie montre que les cultures se trouvent principalement dans le nord du pays en raison des conditions climatiques plus favorables. En effet, le climat du nord du pays se caractérise par des précipitations plus élevées, des températures modérées, des sols fertiles et une meilleure disponibilité en eau pour l'irrigation, soutenant ainsi une végétation plus dense et des zones agricoles.

Les données déséquilibrées (forte proportion de la classe 0 plus particulièrement) constituent un défi majeur en machine learning, car chaque modèle réagit diffé-

remment à ce problème. Par exemple, les modèles basés sur les arbres, comme les arbres de décision, les forêts aléatoires (**Random Forest**) et le gradient boosting (**XGBoost**), ont tendance à avoir un biais faible et une variance élevée en présence de déséquilibre des données. Cela conduit souvent à un surapprentissage (overfitting), car ces algorithmes se concentrent excessivement sur les classes majoritaires. En revanche, les réseaux de neurones, face à ce dilemme, tendent à favoriser la classe majoritaire. Ces modèles nécessitent de grandes quantités de données variées pour fonctionner efficacement. En conséquence, ils deviennent biaisés envers la classe majoritaire et, dans des cas extrêmes, peuvent complètement ignorer la classe minoritaire [9].

Dans notre cas, nous disposons d'un grand nombre de données (349403 lignes), nous avons donc opté pour la deuxième solution en utilisant le sous-échantillonnage aléatoire (**Random Under-Sampling, RUS**) [3]. Son application conduit à la nouvelle répartition suivante :

```
Distribution des classes avant sous-échantillonnage :
[0 : 247828, 1 : 10690, 2 : 21004]
Class percentages: {0: 88.7, 1: 3.8, 2: 7.5}
Distribution des classes après sous-échantillonnage :
[0 : 20000, 1 : 10690, 2 : 21004]
Class percentages: {0: 38.6892, 1: 20.6793, 2: 40.6314}
```

Voici les histogrammes avant et après l'under sampling :

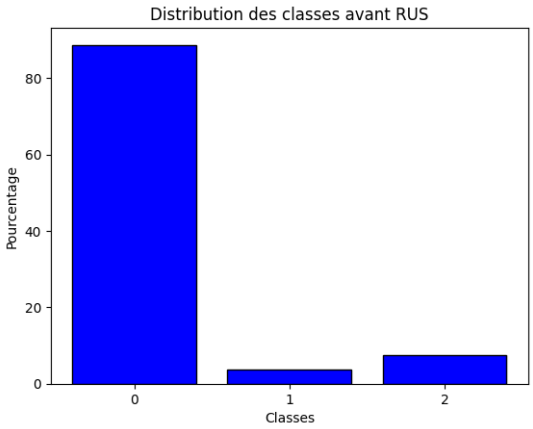

FIGURE 2.6 – Avant undersampling

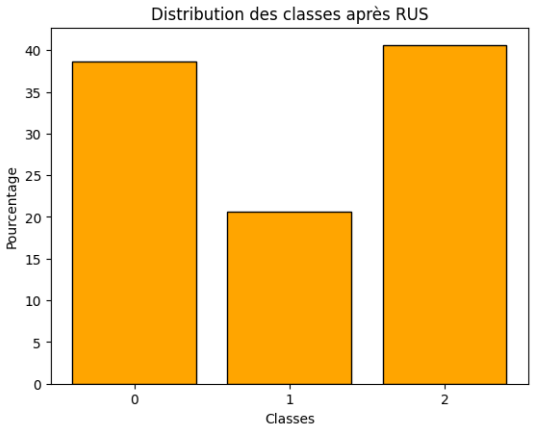

FIGURE 2.7 – Après undersampling

Chapitre 3

Prediction de la couverture forestiere en Tunisie

3.1 Introduction

Après avoir traité les données comme indiqué dans les chapitres précédents et recherché les modèles de classification les plus couramment utilisés dans ce contexte [14], nous avons identifié les suivants : **Random Forest**, **XGBoost**, **SVM** (Support Vector Machine) et **ANN** (Artificial Neural Network). Ces différents modèles ont été construits en utilisant les données décrites ci-dessus en utilisant la bibliothèque Python **sklearn**. Cette bibliothèque contient différents modèles prédéfinis de machine learning , tout en expliquant la formulation mathématique de chaque algorithme afin de choisir le meilleur pour la prédiction de la couverture forestière en Tunisie. Ces différents modèles sont décrits dans les sections suivantes :

3.2 Random Forest Classifier

Le modèle Random Forest, Développé par **Leo Breiman** et **Adele Cutler** [6], est un algorithme d'apprentissage automatique qui utilise un ensemble d'arbres de décision pour améliorer la précision et la robustesse des prédictions.

3.2.1 Aspect théorique

Puisque le modèle Random Forest est composé d'un ensemble d'arbres de décision, il est crucial de comprendre le fonctionnement d'un arbre de décision.
Un arbre de décision est une structure arborescente où chaque nœud interne représente un test sur une caractéristique, chaque branche représente le résultat du test, et chaque feuille représente une décision finale ou une prédiction.

L'algorithme pour construire un arbre de décision se manifeste par le partitionnement de manière récursive de l'espace des fonctionnalités D en sous-ensembles basés sur les valeurs des caractéristiques, de telle sorte que les échantillons avec les mêmes étiquettes ou des valeurs cibles similaires soient regroupés. Le processus commence à la racine de l'arbre, où le jeu de données est divisé en deux sous-ensembles selon la caractéristique qui minimise l'impureté. Cette division se répète de manière récursive sur chaque sous-ensemble jusqu'à ce qu'une condition d'arrêt soit atteinte. Les feuilles

de l'arbre correspondent aux décisions finales ou aux prédictions.

Voici le déroulement de l'algorithme étape par étape [13] :
- Soit $D = (X_1, \ldots, X_n)$ et Y notre jeu de données où les $X_i, i \in (1, \ldots, n)$ sont les caractéristiques de formation (prédicteurs) et Y le vecteur d'étiquettes (réponse) que l'on veut prédire.
Supposons par exemple que Y contient m classes, c'est-à-dire :
$\forall i \in (1, \ldots, n)$, $y_i \in (1, \ldots, m)$.
- On pose p_k , $k \in (1, \ldots, m)$: la proportion des instances appartenant à la classe k dans le vecteur Y.

— **Étape 1 : Calcul de l'impureté :**
L'impureté mesure la diversité des classes dans un ensemble de données. Les mesures courantes sont l'entropie, l'indice de Gini et l'erreur de classification :
- Indice de Gini : Gini(D) $= 1 - \sum_{k=1}^{K} p_k^2$
Entropie : $Entropy(D) = -\sum_{k=1}^{K} p_k \log(p_k)$
Erreur de classification : $Error(D) = 1 - \max(p_1, p_2, \ldots, p_K)$

— **Étape 2 : Choix de la caractéristique et du seuil :**
Pour chaque caractéristique X_i et chaque valeur seuil possible t, on divise le jeu de données D en deux sous-ensembles :
- $D_{left} = \{x \in D \mid x_i \leq t\}$
- $D_{right} = \{x \in D \mid x_i > t\}$
On calcule l'impureté totale pour cette division :
- $Impurity(X_i, t) = \frac{|D_{left}|}{|D|} \cdot Impurity(D_{left}) + \frac{|D_{right}|}{|D|} \cdot Impurity(D_{right})$
On choisit la caractéristique et le seuil qui minimisent cette impureté, car finalement on veut des feuilles qui contiennent chacune une classe pure pour faire la prédiction.

— **Étape 3 : Division du jeu de données :**
On divise le jeu de données D en deux sous-ensembles D_{left} et D_{right} en utilisant la caractéristique X_i et le seuil t choisis.

— **Étape 4 : Répétition :**
Les étapes ci-dessus sont répétées de manière récursive pour chaque sous-ensemble D_{left} et D_{right} jusqu'à ce qu'une condition d'arrêt soit atteinte, telle que :
  — Tous les exemples appartiennent à la même classe.
  — Un nombre maximal de niveaux de profondeur de l'arbre est atteint.
  — Un nombre minimal d'instances par nœud est atteint.

La méthode par laquelle le modèle crée plusieurs sous-échantillons aléatoires de notre ensemble de données pour entraîner chaque sous-modèle sur un échantillon, puis choisit le résultat ayant obtenu le plus de votes, est appelée "Bagging". Il s'agit d'une méthode d'apprentissage ensembliste parallèle.
Donc, l'algorithme de Random Forest [10] regroupe plusieurs arbres de décision en attribuant à chaque arbre un échantillon aléatoire D_k à partir de notre jeu de données initial D. Ensuite, pour prédire la classe finale, on choisit la classe majoritaire prédite par tous les arbres.
Mathématiquement, cela se formule comme suit :

- $h(x) = \arg\max_{1 \leq c \leq m} \sum_{q=1}^{Q} \mathbf{1}_{\{h(x,\Theta_q)=c\}}$

où h(x) représente la prédiction finale,c est une classe parmi les m classes possibles, Q est le nombre total d'arbres dans la forêt, et $\mathbf{1}_{\{h(x,\Theta_q)=c\}}$ est la fonction indicatrice qui vaut 1 si l'arbre q prédit la classe c, et 0 sinon.

3.2.2 En pratique

En pratique, j'ai exécuté un modèle **RandomForestClassifier** pour réaliser une classification robuste, j'ai optimisé ses hyperparamètres à l'aide de la commande **"RandomizedSearchCV"**.

Le processus d'optimisation des hyperparamètres consiste à définir une grille de valeurs possibles pour les hyperparamètres et à effectuer une recherche aléatoire sur un certain nombre d'itérations. À chaque itération, une combinaison différente d'hyperparamètres est testée, et le modèle est évalué en utilisant une méthode de validation croisée. Au final, la combinaison d'hyperparamètres qui donne les meilleurs résultats est sélectionnée.

Voici le resultat de l'algorithme obtenu après son éxecution :

```
RandomForestClassifier(bootstrap=False,max_features='log2',
    min_samples_leaf=2,min_samples_split=4,n_estimators=50,
      random_state=0)
```

pour les résultats obtenus avec le modèle de SVM pour la prediction de la couverture forestière actuelle :

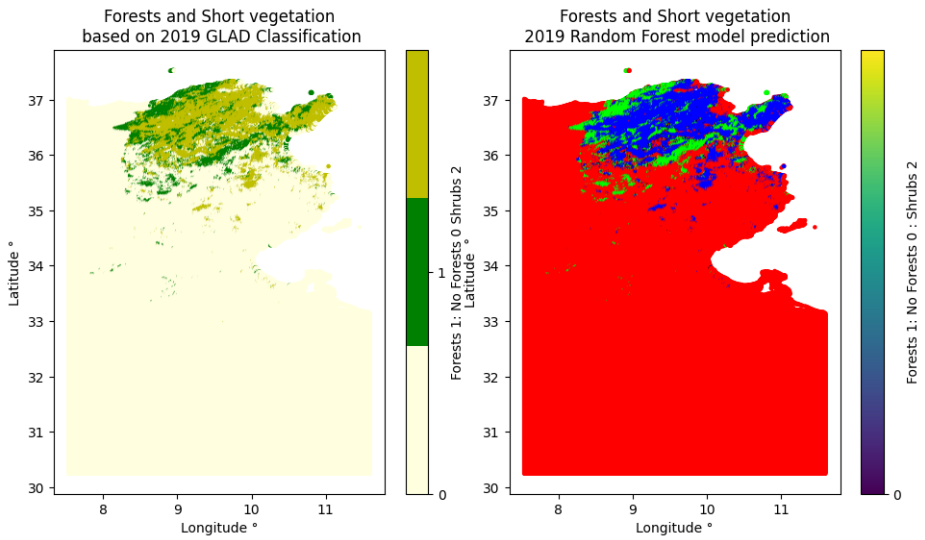

FIGURE 3.1 – Prediction du modèle Random Forest pour la couverture actuelle

et pour la prediction future :

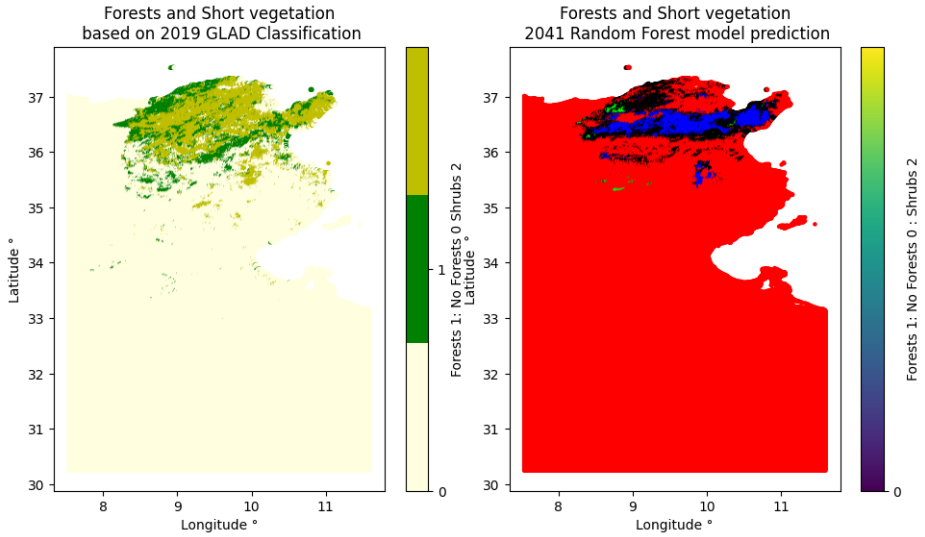

FIGURE 3.2 – Prediction du modèle Random Forest pour la couverture future

3.3 XGBoost classifier

3.3.1 Aspect théorique

Nous allons maintenant utiliser le modèle prédéfini **XGBoost**, ou Extreme Gradient Boosting, est un algorithme de machine learning puissant et efficace, principalement utilisé pour les tâches de classification et de régression. Initialement développé par **Tianqi Chen**, cet algorithme combine de nombreux modèles faibles, généralement des arbres de décision, pour former un modèle robuste en corrigeant continuellement les erreurs des arbres précédents.
XGBoost repose sur la méthode ensembliste **Boosting** qui se déroule en plusieurs étapes :

- **Étape 1 :** On entraîne le premier arbre de décision sur notre jeu de données, en attribuant des poids égaux à toutes les observations. Si une observation est mal classée, son poids est augmenté.
- **Étape 2 :** L'arbre de décision suivant est ensuite ajouté et entraîné sur les données pondérées, afin de corriger les erreurs du premier modèle. Ce processus est répété jusqu'à ce que toutes les observations soient correctement prédites ou que le nombre maximal d'arbres soit atteint.
- **Étape 3 :** Les prédictions finales sont obtenues en combinant les prédictions pondérées de tous les arbres précédents.

Ce modèle repose sur le même concept que Random Forest en utilisant plusieurs arbres de décision. Cependant, alors que les arbres de décision dans Random Forest sont indépendants les uns des autres, XGBoost utilise des arbres dépendants. Chaque nouvel arbre est ajouté pour améliorer les performances du modèle global en se concentrant sur les erreurs commises par les arbres précédents.

Mathèmatiquement, considérons un classifieur faible initial f_0, on cherche à construire un nouveau classifieur faible f_1 à partir de f_0 en introduisant un terme de

résidu :
- $f_1(x) = f_0(x) + h(x)$

De sorte que f_1 soit plus performant que f_0. En répétant l'opération un certain nombre de fois, disons n, on construit un classifieur final F complexe qui est une combinaison linéaire des f_i, où chacun des f_i est associé à un poids α_i :

$F(x) = \sum_{i=1}^{p} \alpha_i f_i(x)$

Le **Gradient Boosting** est une technique particulièrement puissante dans le cas où la fonction de perte (mesure d'écart entre les valeurs théoriques et les valeurs prédites) est différentiable (ce qui est le cas pour une fonction de perte quadratique). Le principe est le suivant :

1. On initialise le modèle avec une valeur constante $f_0(x)$.
2. Pour chaque itération ($1 \leq m \leq M$), on calcule les **pseudo-résidus** : $r_{im} = -\frac{\partial L(y_i, f_{m-1}(x_i))}{\partial f_{m-1}(x_i)}$ ou x_i sont les variables explicatives et y_i les variables cibles.
3. Un classifieur faible h_m est ajusté sur les données x_i et r_{im} (réponses).
4. On détermine un poids associé à ce classifieur faible : $\gamma_m = \arg\min_\gamma \sum_{i=1}^{n} L(y_i, f_{m-1}(x_i) + \gamma h_m(x_i))$
5. On met à jour le modèle : $f_m(x) = f_{m-1}(x) + \eta \gamma_m h_m(x)$

Avec η le taux d'apprentissage, permettant d'éviter un éventuel sur-apprentissage (principe de régularisation).

les pseudo-résidus permettent de se rapprocher de la solution optimale. L'intérêt du nouveau modèle est donc de capter les observations qui n'ont pas été correctement prédites par les classifieurs précédents [1].

3.3.2 En pratique

En utilisant la commande **XGBClassifier** de la bibliothèque **XGBoost**, qui contient plusieurs modèles de gradient boosting prédéfinis, nous avons optimisé le modèle en recherchant les meilleurs hyperparamètres pour ce problème. Cela nous a permis d'obtenir les résultats suivants :

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
  colsample_bylevel=None, colsample_bynode=None,
  colsample_bytree=0.8, device=None, early_stopping_rounds=None,
  enable_categorical=False, eval_metric='logloss',
  feature_types=None, gamma=0.2, grow_policy=None,
  importance_type=None, interaction_constraints=None,
  learning_rate=0.1, max_bin=None, max_cat_threshold=None,
  max_cat_to_onehot=None, max_delta_step=None, max_depth=10,
  max_leaves=None, min_child_weight=None, missing=nan,
  monotone_constraints=None, multi_strategy=None, n_estimators=100,
  n_jobs=None, num_parallel_tree=None, random_state=None, ...)
```

et pour les donneés test qui permet d'evaluer le modèle :

```
y_pred_xgb = xgb_model.predict(X_test_scaled)
xgb_model.score(X_test_scaled, y_test_one_hot)
```

```
0.9358480846009645
```

Pour vérifier le bon fonctionnement du modèle, on affiche la prédiction réalisée par celui-ci concernant la couverture actuelle :

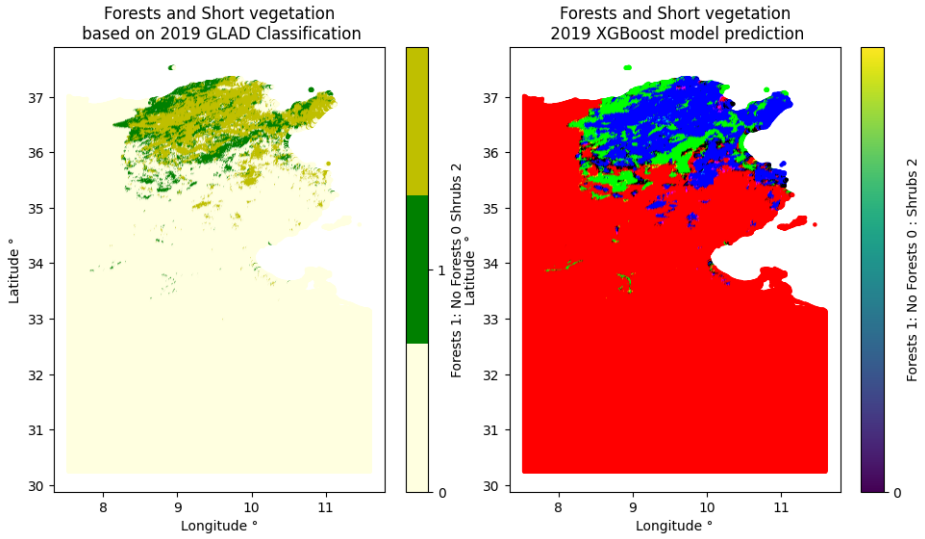

FIGURE 3.3 – Prediction du modèle de XGBoost pour la couverture actuelle

Pour les prévisions futures, avec le scénario SSP 245 pour obtenir les variables bioclimatiques futures. Voici le résultat fourni par le modèle :

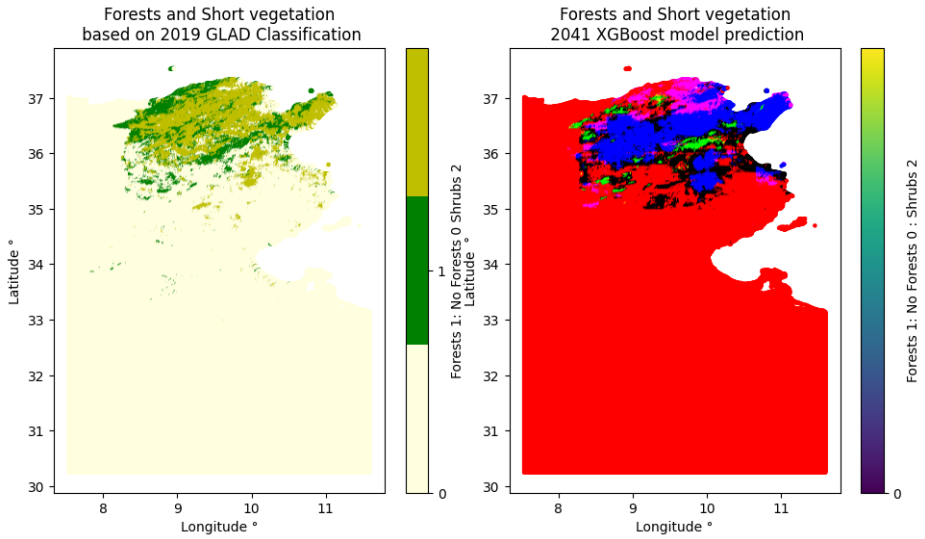

FIGURE 3.4 – Prediction du modèle de XGBoost pour la couverture future

3.4 SVM (Support Vector Machine)

3.4.1 Théorie

Le Support Vector Machine **(SVM)** est un algorithme d'apprentissage supervisé utilisé pour la classification et la régression. Le but principal d'un **SVM** est de trouver l'hyperplan optimal qui sépare les classes de données de manière à maximiser la marge entre les classes [15].

Tout d'abord, il est utile de rappeler le problème de discrimination linéairement séparable, cad lorsqu'il existe une fonction de décision linéaire, de la forme ($D(x) = \text{sign}(f(x))$ avec $f(x) = v^T x + a$, $v \in \mathbb{R}^p$ et a $\in \mathbb{R}$, classant correctement toutes les observations de l'ensemble d'apprentissage ($D(x_i) = y_i$, $i \in [1, n]$). La fonction f est appelée fonction caractéristique.
Nous allons ensuite généraliser au cas des observations non séparables et non linéaires par l'introduction de variables d'écart et de noyaux.

- Formulation du Problème dans le cas linéaire séparable

Pour un problème de classification binaire, nous avons un ensemble de données d'entraînement (x_i, y_i) où $x_i \in \mathbb{R}^n$ représente les caractéristiques et $y_i \in \{-1, +1\}$ les étiquettes de classe.

La distance entre l'hyperplan et le point de données le plus proche est appelée la marge, sa formule est :

$$m = \min_{i \in [1,n]} \text{dist}(x_i, \Delta(v, a))$$

L'objectif est de trouver un hyperplan qui sépare les points des deux classes avec une marge maximale.
L'hyperplan est défini par une équation de la forme :

$$w \cdot x + b = 0$$

où w est un vecteur de poids et b est le biais.

En introduisant explicitement la marge comme une variable, ce problème se réécrit comme un problème d'optimisation sous contraintes :

$$\begin{cases} \max_{v,a} m \\ \text{avec } \min_{i \in [1,n]} \frac{|v^T x_i + a|}{\|v\|} \geq m \end{cases}$$

Ce problème est mal posé car si (v, a) est une solution, (kv, ka) l'est aussi pour tout k > 0. Pour traiter cette difficulté, on effectue le changement de variable : $w = \frac{v}{m\|v\|}$ et $b = \frac{a}{m\|v\|}$.

Le problème se réécrit alors : $\begin{cases} \max_{w,b} \frac{1}{\|w\|} \\ \text{avec } y_i(w^T x_i + b) \geq 1, \quad i = 1, \ldots, n \end{cases}$

La formulation classique des SVM s'obtient alors en minimisant $\|w\|^2$ au lieu de maximiser l'inverse de la norme, ce qui donne le problème suivant, qui admet la même solution que le problème précédent.

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

sous les contraintes :

$$y_i(w \cdot x_i + b) \geq 1 \quad \forall i$$

Ce probléme est quadratique et convexe donc Il admet une solution unique (qui existe puisque le problème est linéairement séparable par hypothèse) et les conditions nécessaires d'optimalité du premier ordre sont aussi suffisantes. Ce problème admet une formulation duale équivalente qui est plus facile a resoudre.

Pour résoudre ce problème d'optimisation avec contraintes, nous introduisons les multiplicateurs de Lagrange $\alpha_i \geq 0$ et formulons le :

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i [y_i(w \cdot x_i + b) - 1] \tag{3.1}$$

Forme Duale

En résolvant le problème primal, nous obtenons la forme duale du problème d'optimisation :

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j)$$

sous les contraintes :

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad , \quad \alpha_i \geq 0$$

Hyperplan Optimal

Les α_i non nuls correspondent aux vecteurs de support. Le vecteur de poids w et le biais b sont obtenus par :

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

Pour calculer le biais b , on utilise l'un des vecteurs de support x_i (pour lesquels $\alpha_i > 0$) :

$$b = y_i - \sum_{j=1}^{N} \alpha_j y_j (x_j \cdot x_i)$$

Classification des Nouveaux Points

Pour un nouveau point x , la classification est déterminée par :

$$f(x) = \text{sign}(w \cdot x + b)$$

SVM Non Linéaire et Kernel Trick

Pour les problèmes non linéaires, nous transformons les données d'origine dans un espace de caractéristiques de plus haute dimension à l'aide de fonctions kernel $K(x_i, x_j)$:

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

Les kernels couramment utilisés sont :

— Polynomial Kernel : $K(x_i, x_j) = (x_i \cdot x_j + 1)^d$

— Gaussian RBF Kernel : $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$

— Sigmoid Kernel : $K(x_i, x_j) = \tanh(\kappa(x_i \cdot x_j) + \theta)$

3.4.2 En pratique

j'ai d'abord initialisé le modèle et déterminé les paramètres de recherche pour l'optimisation des hyperparamètres à l'aide d'une validation croisée. Avec les meilleurs hyperparamètres identifiés, j'ai ré-entraîné le modèle SVM sur l'ensemble de données d'entraînement normalisé. Enfin, j'ai évalué la performance du modèle sur l'ensemble de test. Voici les résultats obtenus :

```
# Paramètres optimaux trouvés
best_params = {'C': 21.830968390524596,'gamma': 0.061737703947045704,
'kernel': 'rbf'}
# Création du modèle SVM avec les meilleurs paramètres
SVM = SVC(C=best_params['C'], gamma=best_params['gamma'],
kernel=best_params['kernel'], probability=True, random_state=42)

# Entraînement du modèle
SVM.fit(X_train_us, y_train_us2)
```

- les résultats obtenus pour la prédiction actuelle :

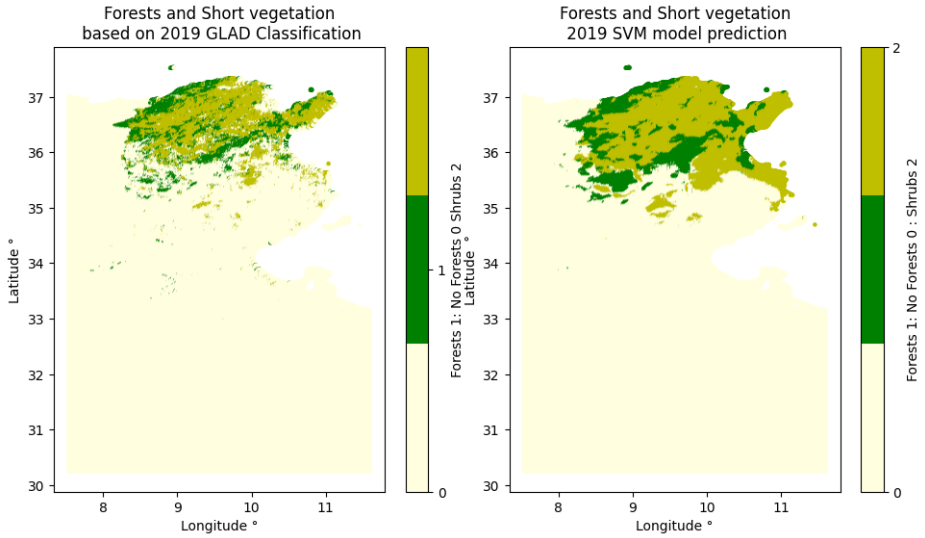

FIGURE 3.5 – Prediction du modèle SVM pour la couverture future

et pour la predicition future :

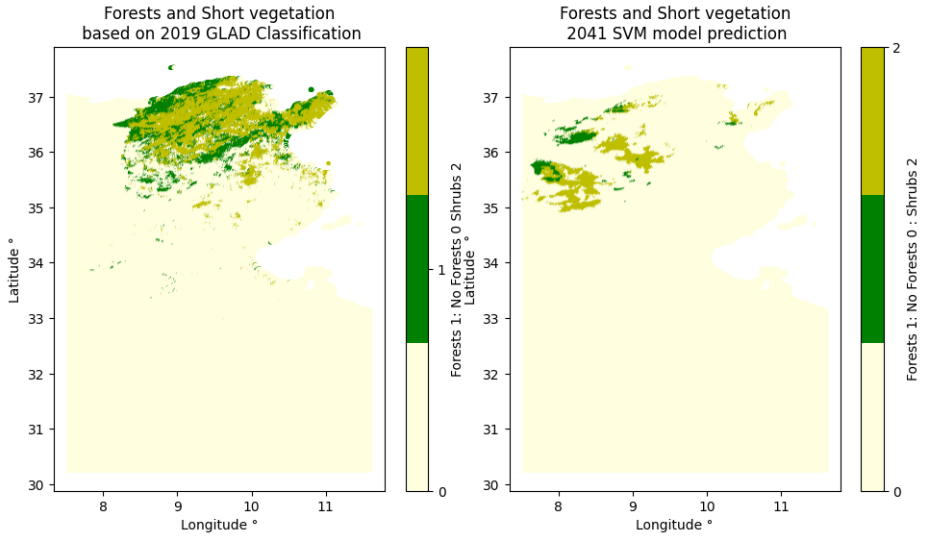

FIGURE 3.6 – Prediction du modèle SVM pour la couverture future

3.5 Réseau de Neurones

3.5.1 Apsect théorique

Les réseaux de neurones artificiels **(ANN)** sont des modèles inspirés du cerveau humain, conçus pour reconnaître des motifs complexes dans les données. Ils sont largement utilisés dans des domaines tels que la classification, la régression, et la reconnaissance de formes. Un réseau de neurones se compose de plusieurs couches de neurones, chaque couche étant entièrement connectée à la suivante. Son processus d'apprentissage repose sur l'ajustement des poids et des biais pour minimiser une fonction de coût qui mesure l'erreur entre la sortie du réseau et la sortie attendue.

On a vu que les modèles linéaires représentent la variable de sortie y comme une fonction linéaire par rapport au vecteur de paramètres w de la variable d'entrée x :
- $y(x, w) := w^T x$.

Les modèles linéaires généralisés tels que la régression logistique utilisent une fonction d'activation non linéaire h :
- $y(x, w) := h(w^T x)$

Les réseaux de neurones sont des compositions itérées (couches) du modèle linéaire généralisé qui se compose typiquement de :

— Une couche d'entrée $x = z^{(0)}$, qui reçoit les données brutes.

— Une ou plusieurs couches cachées, où les calculs sont effectués.

— Une couche de sortie $\hat{Y} = z^{(L)}$, qui produit la prédiction.

Notations

— L : Nombre total de couches (incluant la couche d'entrée et la couche de sortie).

— n_l : Nombre de neurones dans la couche l.

— $W^{[l]}$: Matrice des poids de la couche l, de dimension $n_l \times n_{l-1}$.

— $b^{[l]}$: Vecteur des biais de la couche l, de dimension $n_l \times 1$.

— $a^{[l]}$: Sortie de la couche l, après application de la fonction d'activation.

Voici un schéma qui illustre l'architecture d'un réseau de neurones :

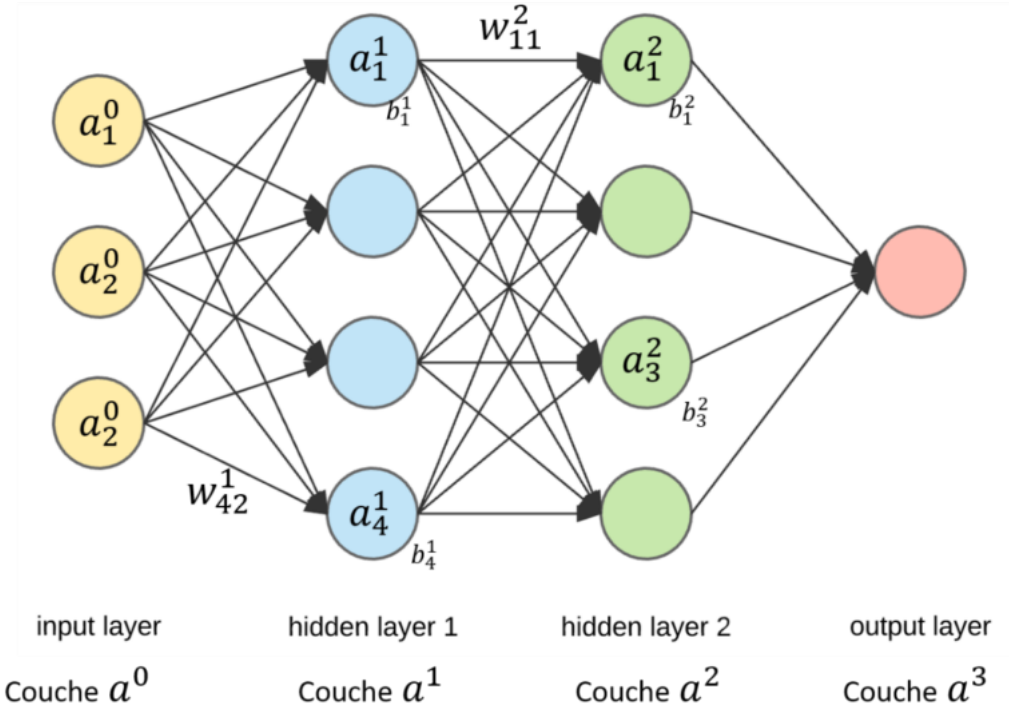

FIGURE 3.7 – Achitecture d'un reseau de neurones [2]

Voici l'algorithme des réseaux de neurones artificiels détaillé :

Étape 1 : Initialisation du Réseau de Neurones :

On doit définir la structure du réseau et initialiser les paramètres.

1. Architecture du réseau :
   — Déterminer le nombre de couches L (incluant la couche d'entrée, les couches cachées, et la couche de sortie).
   — Pour chaque couche l, spécifier le nombre de neurones n_l.
2. Initialisation des poids et des biais :
   — Pour chaque couche l,on initialise la matrice des poids $W^{[l]}$ avec des valeurs aléatoires de petite amplitude (souvent tirées d'une distribution normale ou uniforme).
   — Initialiser les biais $b^{[l]}$ pour chaque neurone de la couche l à zéro ou avec de petites valeurs aléatoires.

Étape 2 : Propagation Avant (Forward Propagation) :

La Propagation Avant (Forward Propagation) est le processus par lequel les données d'entrée traversent successivement les couches d'un réseau de neurones, de la couche d'entrée jusqu'à la couche de sortie afin de calculer la sortie du réseau pour une entrée donnée.

1. Calcul des entrées pondérées :
   — Pour chaque couche l, on calcule l'entrée pondérée $a_j^{(l)}$ de chaque neurone j comme la somme pondérée des sorties $z_i^{(l-1)}$ de la couche précédente $l-1$:
   - $a_j^{(l)} = \sum_{i=1}^{n_{l-1}} w_{ji}^{(l)} z_i^{(l-1)} + b_j^{(l)}$
   où $w_{ji}^{(l)}$ est le poids entre le neurone i de la couche $l-1$ et le neurone j de la couche l.
2. Application de la fonction d'activation :
   — Pour chaque neurone de la couche l, on applique une fonction d'activation $h^{(l)}$ à l'entrée pondérée $a_j^{(l)}$ pour obtenir la sortie $z_j^{(l)}$:
   - $z_j^{(l)} = h^{(l)}(a_j^{(l)})$
   — Exemple de fonctions d'activation : sigmoïde, ReLU, tanh, etc.
3. Répétition :
   — on répéte les étapes 1 et 2 pour chaque couche jusqu'à la couche de sortie, où $z^{(L)}$ donne la prédiction finale $\hat{y}$.

Étape 3 : Calcul de la Fonction de Coût :
La Fonction de Coût Mesure l'écart entre la prédiction du réseau $\hat{y}$ et la valeur réelle y.
Il faut definir une fonction de coût appropriée en fonction de la tâche (par exemple, l'erreur quadratique moyenne pour une régression) :

- $L(w) = \frac{1}{2N} \sum_{n=1}^{N} (\hat{y}_n - y_n)^2$

où N est le nombre d'exemples d'entraînement, $\hat{y}_n$ la prédiction pour le n-ème exemple, et y_n la valeur réelle.

Étape 4 : Rétropropagation (Backpropagation) :

La rétropropagation est l'algorithme utilisé pour minimiser la fonction de coût en ajustant les poids et les biais du réseau.

1. Erreur de la couche de sortie :
   — On calcule l'erreur $\delta_j^{(L)}$ pour chaque neurone de la couche de sortie L :
   - $\delta_j^{(L)} = \frac{\partial L}{\partial a_j^{(L)}} = \hat{y}_j - y_j$
2. Propagation de l'erreur vers les couches précédentes :
   — Pour chaque couche l (en remontant depuis la couche de sortie vers la première couche cachée), calculer l'erreur $\delta_j^{(l)}$ de chaque neurone en fonction de l'erreur de la couche suivante $l+1$.
   - $\delta_j^{(l)} = \sum_{k=1}^{n_{l+1}} \delta_k^{(l+1)} w_{kj}^{(l+1)} h'^{(l)}(a_j^{(l)})$
   où $h'^{(l)}$ est la dérivée de la fonction d'activation.
3. Calcul des gradients des poids :

— Pour chaque couche l, calculer les gradients de la fonction de coût par rapport aux poids $w_{ji}^{(l)}$ et aux biais $b_j^{(l)}$.
- $\frac{\partial L}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} z_i^{(l-1)}$ $\frac{\partial L}{\partial b_j^{(l)}} = \delta_j^{(l)}$

Étape 5 : Mise à Jour des Poids et Biais :
Ajuster les poids et les biais pour réduire l'erreur du réseau en utilisant l'algorithme de descente de gradient :

— Mettre à jour les poids et les biais en utilisant les gradients calculés à l'étape précédente, avec un taux d'apprentissage α.
- $w_{ji}^{(l)} := w_{ji}^{(l)} - \alpha \frac{\partial L}{\partial w_{ji}^{(l)}}$
- $b_j^{(l)} := b_j^{(l)} - \alpha \delta_j^{(l)}$

Étape 6 : Répétition du Processus :
- Répétez les étapes 2 à 5 pour l'ensemble des données d'entraînement jusqu'à la convergence. Cela implique de itérer la propagation avant, le calcul de la fonction de coût, la rétropropagation, et la mise à jour des poids sur plusieurs itérations (appelées époques) jusqu'à ce que la fonction de coût atteigne un minimum satisfaisant ou que le nombre prédéfini d'itérations soit atteint.

3.5.2 En pratique

Durant ce stage, j'ai exploré la possibilité d'utiliser un modèle de réseau de neurones pour la prévision de l'évolution de la couverture forestière en Tunisie. J'ai testé plusieurs configurations d'hyperparamètres, notamment différents optimisateurs comme ADAM, ainsi que diverses architectures de couches et fonctions d'activation. Cependant, les résultats obtenus n'ont pas été à la hauteur des attentes. L'optimisation de ce type de modèle nécessite une maîtrise plus approfondie des réseaux de neurones, et le recours à des techniques de deep learning pourrait également offrir des perspectives plus prometteuses pour ce type de prévision. De plus, d'autres types de réseaux, tels que les réseaux neuronaux récurrents (RNN) ou les réseaux de neurones convolutionnels (CNN), pourraient également être mieux adaptés. En raison du temps limité, je n'ai pas pu approfondir l'étude et l'application de ces concepts. Par conséquent, je recommande que ces aspects soient considérés comme des objectifs pour le prochain stagiaire, afin de poursuivre et d'améliorer ce travail.

3.6 Conclusion

Pour mieux interpréter les prédictions et comprendre les causes, j'ai observé l'importance des variables dans l'entraînement de chaque modèle, à l'exception du modèle SVM. Ce dernier ne fournit pas directement une "importance des variables" comme le font les modèles basés sur des arbres. De plus, pour les SVM non linéaires utilisant des noyaux, cette interprétation devient plus complexe, car la transformation des données dans un espace de caractéristiques de haute dimension complique l'analyse.

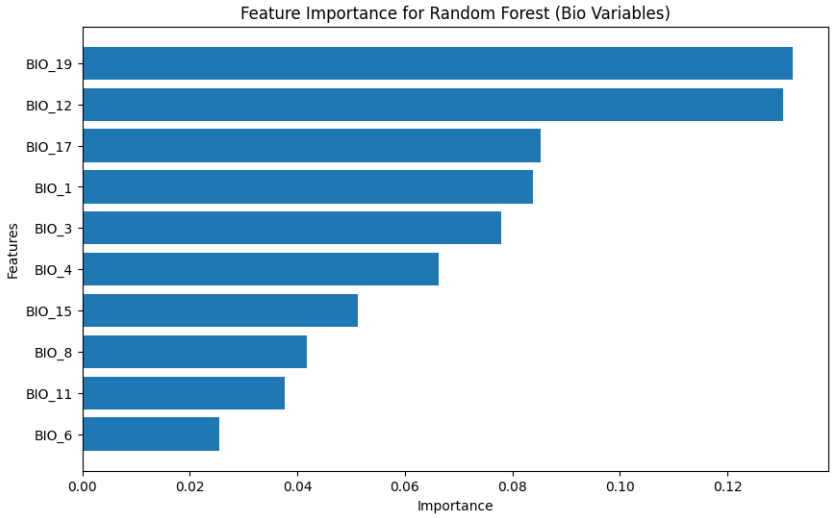

FIGURE 3.8 – Importance des Variables Bioclimatiques pour le Modèle Random Forest

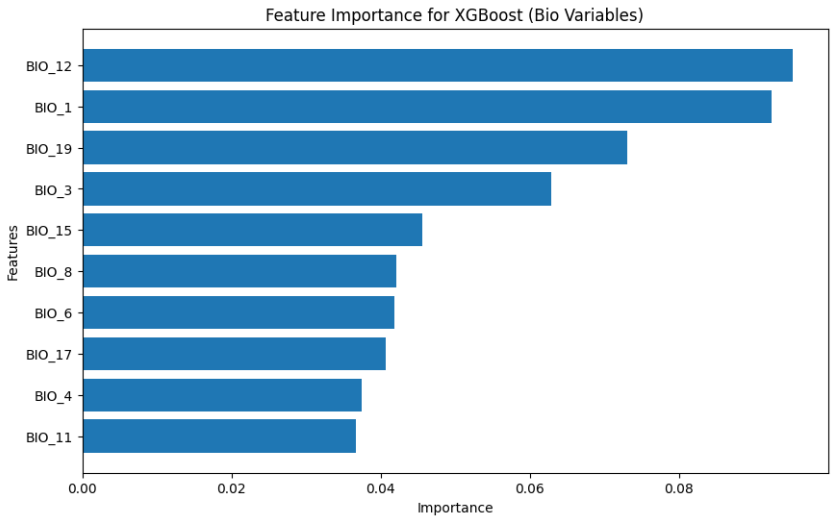

FIGURE 3.9 – Importance des Variables Bioclimatiques pour le Modèle XGBoost

Les prédictions pour 2041, effectuées avec les modèles **XGBoost**, **Random Forest** et **SVM** sous le scénario SSP245, montrent une réduction significative de la couverture forestière en Tunisie, particulièrement dans les régions du nord du pays. Les variables bioclimatiques les plus influentes dans ces prédictions sont les précipitations annuelles (BIO12), les températures moyennes annuelles (BIO1), et les précipitations des trimestres les plus froids (BIO19) et les plus secs (BIO17).

- **XGBoost** identifie BIO12 et BIO1 comme les variables les plus cruciales, soulignant que les variations des précipitations annuelles et des températures moyennes annuelles sont des facteurs déterminants pour la couverture forestière. Les prédictions montrent une forte réduction des zones forestières, avec une fragmentation des zones de végétation persistante.

- **Random Forest** attribue également une grande importance à BIO19 et BIO12, tout en montrant une légère extension des zones de végétation courte par rapport à XGBoost. Ce modèle confirme la réduction importante des forêts.

- **SVM** présente les prédictions les plus pessimistes, avec une réduction drastique des zones forestières et une dispersion plus faible des zones de végétation courte, reflétant une perte sévère de la couverture forestière.

Chapitre 4

Validation des modèles

4.1 Outils de validation

Dans ce chapitre, je vais présenter plusieurs métriques essentielles pour la validation et la comparaison des différents modèles, afin de sélectionner le meilleur.

Pour une classification binaire, où le modèle prédit la classe en fonction d'un seuil déterminé, la matrice de confusion est un outil clé pour évaluer les performances d'un modèle de classification. Ce tableau résume les résultats de prédiction : chaque ligne correspond aux classes réelles, tandis que chaque colonne représente les classes prédites par le modèle. Les cellules situées sur la diagonale principale indiquent les prédictions correctes, les autres cellules mettent en évidence les erreurs de classification, telles que les faux positifs et les faux négatifs. La matrice de confusion permet ainsi de visualiser et de quantifier les erreurs de classification pour chaque classe, offrant une analyse détaillée des performances du modèle, y compris sur des données multiclasses.

Voici une description des différents composants de la matrice de confusion :

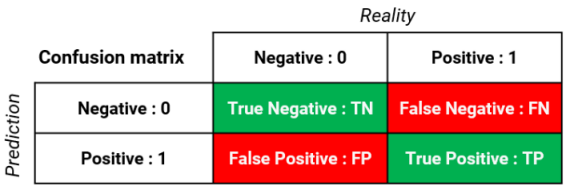

FIGURE 4.1 – Matrice de confusion

- Les Vrais Positifs (TP) : Nombre de fois où le modèle a correctement prédit la classe positive.
- Les Vrais Négatifs (TN) : Nombre de fois où le modèle a correctement prédit la classe négative
. - Les Faux Positifs (FP) : Nombre de fois où le modèle a prédit la classe positive alors que la classe réelle était négative.
- Les Faux Négatifs (FN) : Nombre de fois où le modèle a prédit la classe négative alors que la classe réelle était positive.

À partir de cette matrice de confusion, plusieurs métriques peuvent être calculées pour valider les modèles, telles que :

- **Accuracy** = $\frac{TP+TN}{TP+TN+FP+FN}$: Proportion de prédictions correctes.
- **Precision** = $\frac{TP}{TP+FP}$: Proportion de prédictions positives correctes.

- **Recall** (Rappel) = $\frac{TP}{TP+FN}$: Proportion de vraies valeurs positives correctement identifiées.
- **F1-Score** = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$: Moyenne harmonique de la précision et du rappel.

- **AUC-ROC** courbe (receiver operating caracteristic) : évalue la performance d'un classificateur en traçant le taux de vrais positifs contre le taux de faux positifs pour différents seuils. Elle permet de visualiser la capacité du modèle à distinguer entre les classes positives et négatives. L'aire sous la courbe mesure cette capacité.
Une AUC de 1 indique une performance parfaite, où le classificateur distingue entièrement les classes positives des négatives, une AUC de 0 signifie une inversion complète des classes, tandis qu'une AUC de 0,5 indique une incapacité à différencier les classes, équivalente à des prédictions aléatoires.

- Le **TSS** (True Skill Statistic) : est une mesure importante pour évaluer la performance des modèles de classification. Le TSS prend en compte à la fois les vraies prédictions positives et négatives, offrant une évaluation équilibrée de la performance du modèle sans être influencé par le déséquilibre des classes. Cela le rend plus robuste que d'autres métriques comme l'Accuracy (précision globale), qui peut être trompeuse en présence de classes majoritaires. La formule du TSS pour un problème de classification multiclasses est donnée par :

$$TSS = \frac{\sum_{i=1}^{n_{\text{class}}} (TP_i \cdot TN_i - FP_i \cdot FN_i)}{\sum_{i=1}^{n_{\text{class}}} ((TP_i + FN_i) \cdot (TN_i + FP_i))}$$

Voici le code Python de la fonction TSS que j'ai créée, qui calcule le (True Skill Statistic) moyen entre les classes à partir de la matrice de confusion. Cette fonction est particulièrement utile pour les problèmes de classification multi-classes .

```
    # Calcul du True Skill Statistic (TSS)
def calculate_tss_multiclass(y_true, y_pred):
    classes = np.unique(y_true)
    tss_scores = []
    for cls in classes:
        y_true_bin = (y_true == cls).astype(int)
        y_pred_bin = (y_pred == cls).astype(int)
        tn, fp, fn, tp = confusion_matrix
        (y_true_bin, y_pred_bin).ravel()
        tss = (tp / (tp + fn)) - (fp / (fp + tn))
        tss_scores.append(tss)
    return np.mean(tss_scores)
```

Pour les problèmes de classification multiclasses, il existe trois approches :
- L'approche **"per class"** : S'intéresse à chaque classe indépendamment en transformant chaque classe en classification binaire. Cette approche fournit des synthèses par classe, mais ne permet pas de tirer une synthèse globale sur la robustesse du modèle.
- L'approche **"macro"** : À partir des calculs trouvés dans l'approche "per class", elle permet de résumer et de tirer des synthèses globales pour le modèle. Cette approche a deux méthodes : l'approche classique, robuste aux déséquilibres de classes,elle fait la moyenne des métriques "per class" par la formule génerale suivante :

$$\text{macro-metric} = \frac{1}{n_{\text{class}}} \sum_{i=1}^{n_{\text{class}}} \text{metric}_{\text{class}_i}$$

et une approche pondérée qui représente mieux les données en fesant la moyenne pondérée par la formule suivante :

$$\text{macro-weighted-metric} = \sum_{i=1}^{n_{\text{class}}} \text{prop}_i \cdot \text{metric}_{\text{class}_i}$$

- L'approche **"micro"** : utilise directement les valeurs de la matrice de confusion multiclasses pour produire une métrique qui résume la performance du modèle. Dans notre cas, il est préférable d'utiliser l'approche "macro" classique, car nous avons le problème du déséquilibre des données de test [8].

4.2 Evaluation des modèles avec les donneés tests

Après avoir entraîné les différents modèles sur les données d'entraînement, passant maintenant au évaluation en utilisant les données de test afin de choisir le meilleur prédicteur.

Voici les résultats du modèle Random Forest :

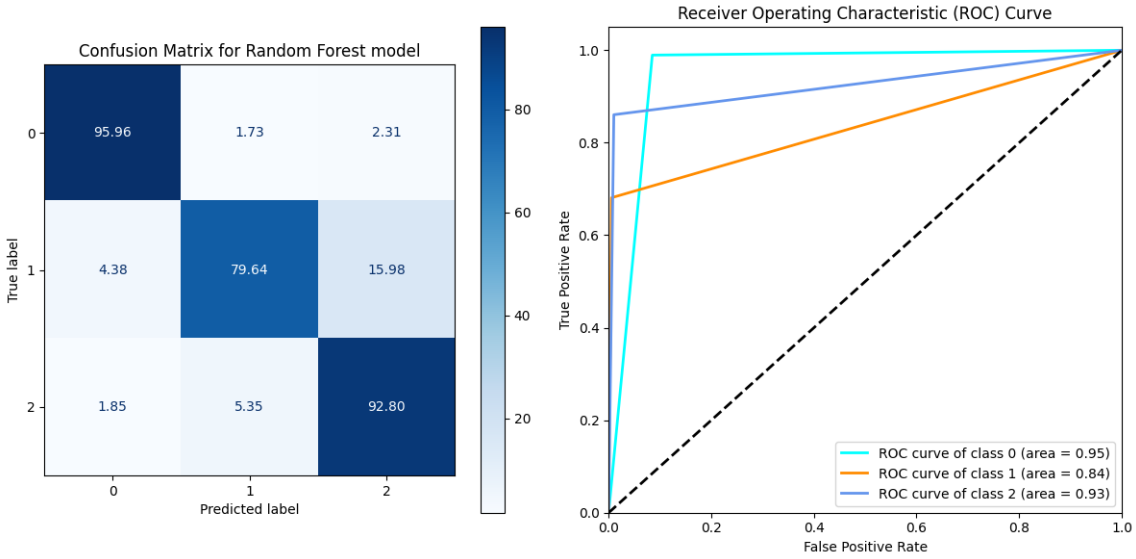

FIGURE 4.2 – Matrice de confusion et ROC courbe pour le modèle de Random Forest aprés l'undersampling

La figure montre que le modèle de forêt aléatoire prédit correctement les classes 0 et 2 avec des précisions respectives de 95,96 % et 92,80 %, tandis que la précision pour la classe 1 est de 79,64 %, ce qui reste acceptable compte tenu du nombre d'échantillons de cette classe. La courbe ROC indique une excellente capacité de discrimination pour les classes 0 et 2 avec des AUC de 0,95 et 0,93, et une bonne capacité pour la classe 1 avec une AUC de 0,84.

Pour évaluer l'utilité de la méthode d'undersampling pour gérer le déséquilibre des données, voici les résultats obtenus par le modèle sur les données initiales :

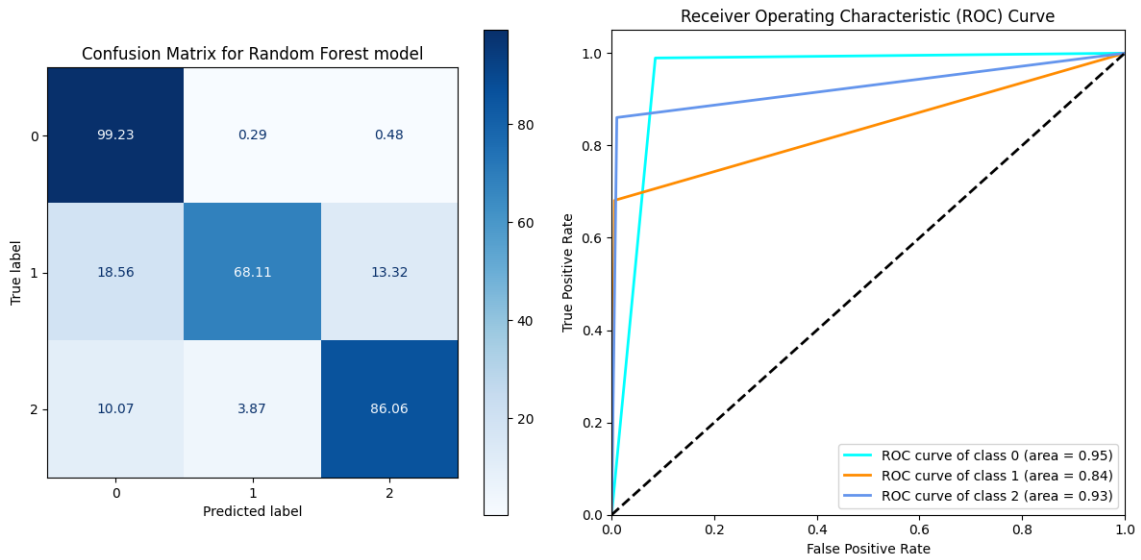

FIGURE 4.3 – Matrice de confusion et ROC courbe pour le modèle de Random Forest avant l'undersampling

Comme l'indiquent les deux figures précédentes, les prédictions correctes pour les deux classes minoritaires ont élevées, tandis que celles pour la classe majoritaire ont diminué. Cela montre un compromis nécessaire, car notre objectif principal est de détecter les couvertures forestières et les cultures futures. Cela souligne l'importance d'utiliser des méthodes d'undersampling pour améliorer la performance du modèle.

Pour le modèle XGBoost :

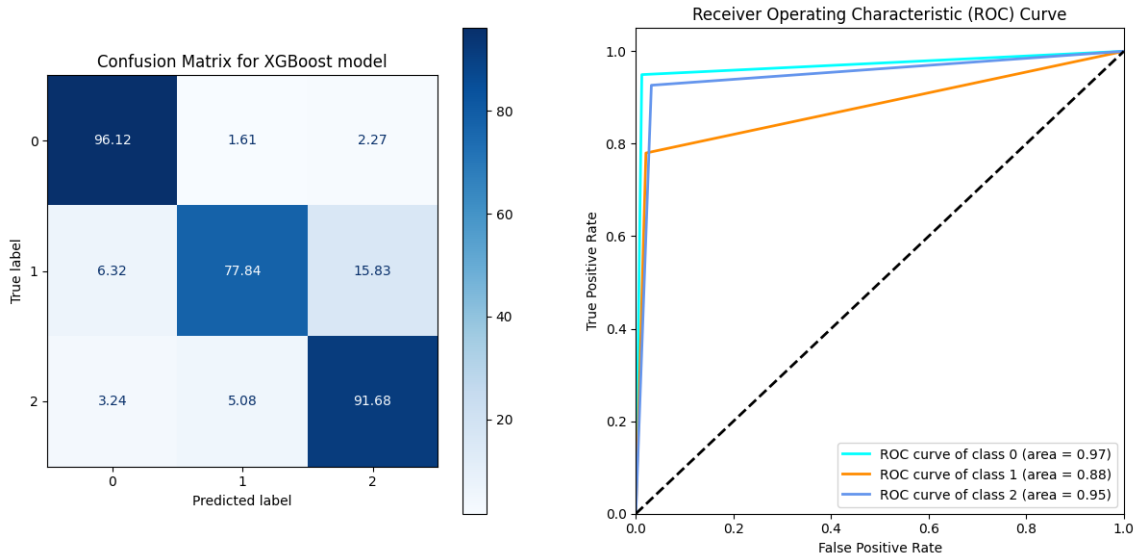

FIGURE 4.4 – Matrice de confusion et ROC courbe pour le modèle de XGBoost

Le modèle XGBoost montre une meilleure performance globale que le modèle Random Forest. En termes de précision, XGBoost surpasse légèrement Random Forest pour les classes 0 et 2, tandis que Random Forest a une précision légèrement supérieure pour la classe 1. Les courbes ROC indiquent également une meilleure performance pour XGBoost, avec des AUC de 0.97, 0.88, et 0.95 pour les classes 0,

1, et 2 respectivement, contre 0.95, 0.84, et 0.93 pour Random Forest. Globalement, XGBoost semble plus performants.

Voici maintenant les résultats fournis par le modèle SVM pour la matrice de confusion :

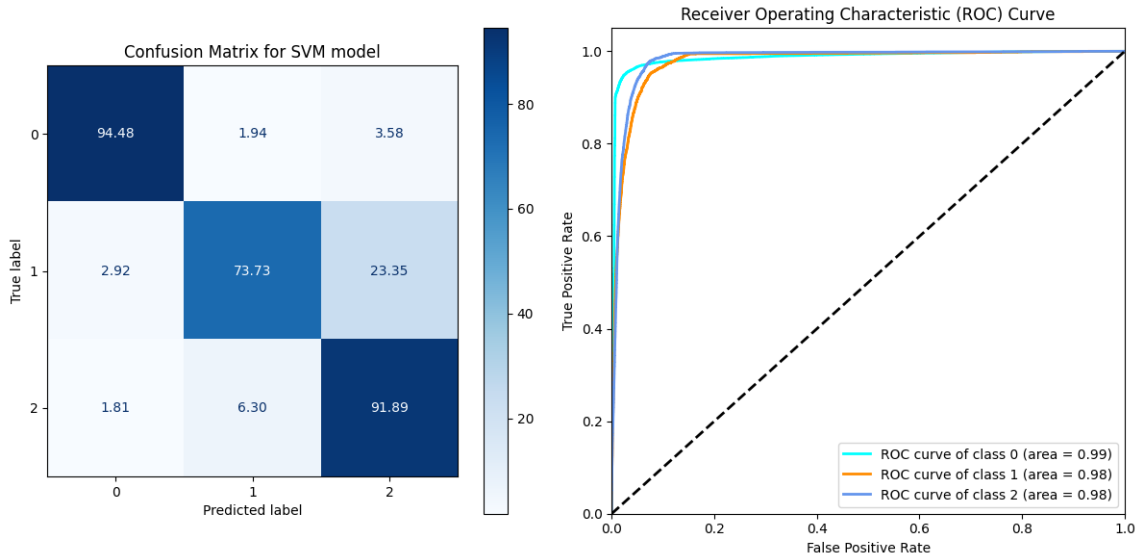

FIGURE 4.5 – Matrice de confusion et ROC courbe pour le modèle SVM

Le modèle SVM montre une bonne performance globale avec une AUC élevée (0.98-0.99) pour toutes les classes, mais présente une presicion notable a prédire les classes 0 et 2, mais de précision faible pour la classe 1 (73.73%). En comparaison, le modèle Random Forest offre une meilleure précision pour toutes les classes, surtout pour la classe 1 (79.64%), mais avec une AUC légèrement inférieure (0.84) pour cette classe. Le modèle XGBoost se distingue en combinant une bonne précision, proche de Random Forest, avec des AUC élevées, similaires à celles du SVM, ce qui en fait un choix légèrement supérieur, notamment pour sa capacité à mieux distinguer les classes dans des situations où la classe 1 est cruciale.

4.3 Conclusion

Pour choisir le meilleur modèle, il est crucial d'examiner les métriques à l'aide d'un tableau qui résume les métriques moyennées sur les trois classes pour chaque modèle. Ce tableau, associé aux matrices de confusion de la section précédente, aidera à déterminer le meilleur modèle. Voici le tableau récapitulatif suivant :

Modèle	ROC AUC moyen	Précision	Rappel	F1 score	TSS
RF	0.9366	0.7770	0.8946	0.8274	0.8693
XGBoost	0.9333	0.7801	0.8854	0.8260	0.8554
SVM	0.98	0.7295	0.8669	0.7850	0.8374

Pour choisir le modèle le plus performant, on a comparé les métriques résumées dans le tableau et analysé les matrices de confusion. Le modèle Random Forest offre un bon équilibre avec une précision de 77.70% et un rappel de 89.46%, bien qu'il ait une AUC légèrement inférieure (0.9366). Son F1 score est de 0.8274, et son TSS est le plus élevé (0.8693), indiquant une excellente capacité à discriminer entre les classes. XGBoost montre une performance globale precieuse avec des AUC élevées (0.97, 0.88, 0.95) et une précision améliorée (78.01%) pour les classes 0 et 2, bien que légèrement inférieure pour la classe 1. Son F1 score est de 0.8260, et son TSS de 0.8554 est légèrement inférieur à celui de Random Forest, mais reste performant tandi que SVM affiche des AUC très élevées (0.98-0.99), mais une précision plus faible (72.95%) et des performances particulièrement faibles pour la classe 1 (73.73%), avec un F1 score de 0.7850 et un TSS de 0.8374, le plus bas parmi les trois modèles. En combinant une bonne précision, un rappel global, des AUC élevées, un F1 score et un TSS solide, Random Forest et XGBoost se distinguent comme les modèles les plus performants, avec un léger avantage pour Random Forest, notamment dans les situations où la distinction entre les classes est cruciale, particulièrement pour la classe 1.

Conclusion

En conclusion, ce stage m'a permis de développer et évaluer plusieurs modèles de machine learning pour prédire la couverture forestière en Tunisie pour l'année 2041, en utilisant des variables bioclimatiques sous le scénario SSP245. Les modèles évalués incluent Random Forest, XGBoost et SVM. Après une analyse approfondie des matrices de confusion et des métriques robustes telles que la précision, le rappel, le F1 score, l'AUC et le TSS, les modèles Random Forest et XGBoost se sont distingués comme les plus performants. Random Forest, en particulier, a montré un bon équilibre entre précision et rappel, ainsi qu'une capacité discriminante supérieure, indiquée par le TSS le plus élevé.

Les prédictions effectuées sous le scénario SSP245, même optimiste, révèlent une réduction dramatique des zones forestières, surtout dans le nord de la Tunisie. Cette diminution de la couverture forestière souligne l'urgence d'adopter des mesures de conservation robustes pour préserver ces écosystèmes vitaux. Il est crucial de mettre en place des politiques de gestion durable, et des pratiques agricoles respectueuses de l'environnement pour atténuer les impacts du changement climatique. Ces actions sont essentielles pour garantir la préservation de nos forêts pour les générations futures.

Bibliographie

[1] Blent.ai. Xgboost : tout comprendre. https://blent.ai/blog/a/xgboost-tout-comprendre, 2024. Consulté le 29 juillet 2024.

[2] N. Combari. Réseau de neurones : On va essayer de démystifier un peu tout ça ! https://www.aspexit.com/reseau-de-neurones-on-va-essayer-de-demystifier-un-peu-tout-ca-1/, 2020. Consulté le 12 août 2024.

[3] C. Ellis. Oversampling vs undersampling : Which technique is better ? https://crunchingthedata.com/oversampling-vs-undersampling/, 2023. Accessed : 2024-06-20.

[4] S. E. Fick and R. J. Hijmans. Worldclim 2 : New 1-km spatial resolution climate surfaces for global land areas. https://worldclim.org/data/worldclim21.html, 2017. Accessed : 2024-06-12.

[5] M. e. a. Hansen. Global land cover and land use 2019 strata file. https://storage.googleapis.com/earthenginepartners-hansen/GLCLU_2019/strata.txt, 2019. Accessed : 2024-06-12.

[6] IBM. Random forest. = https ://www.ibm.com/fr-fr/topics/random-forest, n.d. Consulté le 14 août 2024.

[7] IGAD Climate Prediction and Applications Centre (ICPAC). Geoportal icpac layer : afr_g2014_2013_0. https://geoportal.icpac.net/layers/geonode%3Aafr_g2014_2013_0, 2013. Accessed : 2024-06-12.

[8] D. Kobia. Classification metrics for multi-class simple. https://kobia.fr/classification-metrics-multi-class-simple/, 2024. Accessed : 2024-07-21.

[9] J.-M. Kobia. Imbalanced data et machine learning. https://kobia.fr/imbalanced-data-et-machine-learning/, 2024. Accessed : 2024-06-19.

[10] M. Kon. Random forests : A mathematical perspective. https://math.bu.edu/people/mkon/MA751/L19RandomForestMath.pdf, Year. Accessed : Date.

[11] QGIS Development Team. *QGIS User Manual*, 2024. Accessed : 2024-06-13.

[12] D. N. Science. Managing highly correlated features for machine learning. https://datanscience.com/managing-highly-correlated-features-for-machine-learning/, 2023. Accessed : 2024-06-18.

[13] Scikit-learn Contributors. Decision trees in scikit-learn. https://scikit-learn.org/stable/modules/tree.html. Accessed : 2024-06-27.

[14] Sowmya. A review on the work of crop yield prediction. *BioGecko*, 2024. Accessed : 2023-03-5.

[15] Unknown. Svm (support vector machine). `https://blent.ai/blog/a/svm-support-vector-machine`, 2023. Consulté le 31 juillet 2024.

[16] A. Vidhya. Feature scaling in machine learning : Normalization vs. standardization. *Analytics Vidhya Blog*, 2020. Accessed : 2024-06-24.

[17] WorldClim. Worldclim cmip6 climate data. `https://www.worldclim.org/data/cmip6/cmip6_clim30s.html`, ongoing.