Kéhina Manseri
Alix Sirven-Viénot

Fouille de texte
Yoann Dupont

RAPPORT

Entraînement de classifieurs par apprentissage automatique et comparaison des performances de différents algorithmes de classification sur une tâche de reconnaissance de compatibilité de recettes avec des régimes alimentaire

Lien du GitHub:
https://github.com/KehinaleK/fouille_de_textes

Introduction

Nous avons choisi ce thème car nous aimons partager des moments conviviaux autour de repas entre amis ou en famille. En discutant toutes les deux, le sujet des restrictions alimentaires est venu naturellement sur la table. Lorsque nous avons dû choisir un projet de fouille de texte, il nous tenait à cœur de trouver un sujet porteur de sens, ayant un intérêt concret. Ce projet pourrait être utilisé dans de nombreux contextes (que nous aborderons en conclusion).
Nous souhaitions revenir en premier lieu sur la notion de régime. À ne pas confondre avec la notion de sain ou d'équilibre. Nous avons donc sélectionné des régimes restrictifs qui peuvent être choisis ou subis par les personnes les suivant. Les régimes alimentaires sont des sujets très personnels touchant des personnes dans leur intimité et dans leurs convictions morales ou religieuses. Ce n'est pas une notion qui peut se résumer en binaire car ce qui est bon pour un ne le sera pas forcément pour un autre. Ne pouvant donc choisir une unique catégorie binaire, oui/non ou sain/mauvais, nous avons décidé d'entraîner la machine sur plusieurs régimes alimentaires basés sur des convictions personnelles ou des restrictions alimentaires (allergie ou intolérance . Nous testerons ainsi sa capacité à repérer différents régimes.

Les régimes choisis sont les suivants :

- Végétarien
- Végan
- Crudivore
- Sans gluten
- Alix proof
- Sans noix

Nous avons choisi ces régimes car ce sont les régimes les plus courants en France et pour leur diversité d'attributs. De par leur attribut divers, chacun de ces régimes présentait des challenges différents pour l'apprentissage automatique.

Répartition des tâches:
Pour ce projet, il n'y à pas eu de répartition formelle des tâches ce qui explique que nous n'ayons qu'une branche main dans notre git. En effet pour chaque étape du projet nous avons travaillé côte à côte en liveshare.

Sommaire

1. Création du corpus

1.1. Obtention des autorisations

Nous avons envoyé des emails à des personnes tenant des blogs de cuisine afin de leur demander l'autorisation d'utiliser leurs recettes.
Nous avons reçu trois réponses positives que vous trouverez ci-dessous:

- Jacqueline Mercorelli aussi appelé Mercotte du site *La Cuisine de Mercotte*, [1]
- Jackie Thouny qui tient le site *La Cuisine de Jackie*,[2]
- Nadine Thomas qui tient le site *Nad cuisine*.[3]

Mail de Jackie Thomas:

Bonjour Alix et Kehina,

Tout d'abord je vous remercie de l'intérêt que vous portez à mon blog, il représente beaucoup pour moi.
Votre projet est une démarche très intéressante et je vous en félicite,
Je vous autorise tout à fait à piocher dans les recettes du blog pour travailler à l'élaboration de votre alogorithme de recherche.
Je vous souhaite bonne chance dans la réalisation de votre projet d'étude.
Bine à vous.
Jackie

Mail de Mercotte:

Bonjour Kehina,
Merci pour votre mail et pas de problème pour que certaines recettes fassent partie de votre corpus sans être publiées et que les références soient faites correctement !
Bon courage
Bien à vous
Mercotte

Mail de Nadine :

Bonjour,

Oui bien sûr vous pouvez utiliser certaines de mes recettes (contente qu'elles vous plaisent d'ailleurs!). Merci d'avoir demandé avant, peu ont cette politesse.
Bon courage pour votre projet!

Nadine **THOMAS**

Pour compléter ce corpus nous avons utilisé des sites autorisant la reproduction de leur contenu comme le site *Marmiton.* Ci-dessous vous trouverez **les mentions légales de *Marmiton***.

PROPRIETE INTELLECTUELLE

L'intégralité des pages, textes, images, graphiques, animations, vidéos, sons et autres œuvres ainsi que leurs arrangements représentés sur le Site sont protégés par droit d'auteur ou d'autres droits de protection prévus par la loi sont déposés par le Groupe Reworld Media. Sauf accord explicite, aucune licence n'est accordée au titre du présent Site.

Il est interdit de copier, diffuser ou modifier tout ou partie du contenu du présent Site à des fins commerciales ou autre et d'en permettre l'accès à des tiers sans l'autorisation préalable écrite de l'Editeur. Nous attirons l'attention sur le fait que le Site contient des images pouvant être soumises au droit d'auteur de tiers.

[1] https://www.mercotte.fr
[2] https://www.jackiecuisine.com/
[3] https://quandnadcuisine.fr/

1.2. Extraction des recettes et catégorisation

La première étape nécessaire pour la constitution de notre corpus consistait à lister un nombre conséquent de recettes pouvant être catégorisées dans les régimes choisis et ainsi permettre l'entraînement de nos modèles.
Le listing de ces recettes a été effectué manuellement afin d'essayer d'obtenir un nombre minimal de recettes pour chaque régime. Ces dernières étaient répertoriées dans un document Google Sheet se présentant de la manière suivante :

id	liens	vegetarien	vegan	crudivore	sans_gluten	alixproof	sans_noix	sucré/salé
1	https://quandnadcuisine.fr/salade-de-quinoa-a-la-mexicaine/	1	1	0	1	1	1	salé
2	https://quandnadcuisine.fr/bricks-au-thon-et-tomates-sechees/	0	0	0	0	0	1	salé
3	https://quandnadcuisine.fr/ballotins-au-chevre-et-au-bacon/	0	0	0	0	0	1	salé
4	https://quandnadcuisine.fr/clafoutis-petits-pois-et-surimi/	0	0	0	0	0	1	salé
5	https://quandnadcuisine.fr/clafoutis-poireaux-chorizo-a-la-multidelices/	0	0	0	1	0	1	salé
6	https://quandnadcuisine.fr/pesto-de-fanes-de-radis/	1	0	1	0	0	0	salé
7	https://quandnadcuisine.fr/tartines-figues-et-chevre/	1	0	0	0	0	0	salé
8	https://quandnadcuisine.fr/salade-de-pommes-de-terre-au-citron/	1	1	0	1	0	1	salé
9	https://quandnadcuisine.fr/terrine-de-courgettes-au-thon-et-au-chevre-frais/	0	0	0	1	0	1	salé
10	https://quandnadcuisine.fr/veloute-de-navets-et-patates-douces-aux-epices-tandoori/	1	0	0	1	0	0	salé
11	https://quandnadcuisine.fr/veloute-de-courge-et-patate-douce/	1	1	0	1	0	1	salé
12	https://quandnadcuisine.fr/veloute-de-potiron-au-kiri/	0	0	0	1	0	1	salé

Nous disposons d'une colonne ***id*** permettant d'attribuer un identifiant unique à chaque recette, d'une colonne ***liens*** contenant les liens de la page correspondante à chaque recette et enfin de 6 colonnes permettant d'attribuer une valeur de ***0*** ou ***1*** à chaque régime. Si la recette concernée était par exemple bien ***vegan***, un 1 était alors placé dans la colonne ***vegan***. Un 0 était placé dans les colonnes des régimes auxquels ne répondait pas la recette. Enfin, nous disposons également d'une colonne ***sucré/salé*** permettant d'obtenir des informations supplémentaires utiles à l'entraînement de nos modèles et à l'établissement de bilans statistiques. La structure de ce tableau suit donc une représentation de données en ***one-hot encoding*** nous permettant notamment d'entraîner nos modèles sur un même document mais en fonction de valeurs de prédiction différentes, ici en fonction de régimes cibles différents. De plus, nous avions appliqué une mise en forme conditionnelle au document permettant de surligner les lignes en double et nous assurant ainsi une corpus composé de recettes uniques.

1.3. Traitement du corpus

Une fois l'ensemble des liens et des valeurs pour chaque régime obtenus, nous avons pu commencer à extraire le contenu textuel de chaque page afin d'isoler les textes pertinents pour l'entraînement de nos modèles.

retrieval.py :

Le script ***retrieval.py*** permet d'extraire l'ensemble des contenus textuels des pages répertoriées dans notre fichier tabulaire (précédemment importé en format ***csv*** du document ***Google Sheet***.)

```
def get_table():
    tableau = pd.read_csv("liens.csv")
    colonne_liens_id = tableau[["id", "liens"]]
    return colonne_liens_id

def get_liens(colonne_liens_id, debut, fin):
```

```
    lignes_necessaires = colonne_liens_id[debut:fin]
    lignes_necessaires["id"] = lignes_necessaires["id"].astype(int)
    liste_liens = list(lignes_necessaires.itertuples(index=False,  name=None))
    return liste_liens
```

Nous utilisons la librairie ***Pandas*** afin d'extraire de notre fichier tabulaire une liste de ***liens*** et leurs ***id*** correspondant. Nous utilisons ensuite la libraire ***BeautifulSoup*** afin d'extraire le contenu textuel de chaque lien.

```
def get_texte(liste_liens, site):
    for id, lien in liste_liens:
        url = lien
        reponse = requests.get(url)
        soupe = BeautifulSoup(reponse.text, "html.parser")
        texte = soupe.get_text()
        chemin = f"dumps-text/{site}/{id}_{site}_dump.txt"
        with open(chemin, "w", encoding="utf8") as fichier:
            fichier.write(texte)
```

Ces contenus textuels sont ensuite sauvegardés dans des fichiers ***txt*** placés dans des dossiers nommées d'après les sites de publications eux-même placés à l'intérieur d'un dossier dumps-text. Ce nommage a pu être réalisé à l'aide de l'ordre des recettes précédemment structuré en fonction des sites de publication et l'intégration d'un gestionnaire d'arguments dans le programme.

```
def main():
    parser = argparse.ArgumentParser(description='Extraire le contenu textuel')
    parser.add_argument('site', choices=["nadine", "jackie", "mercotte", "marmiton",
"elle"],help='Nom du site dont on veut extraire les liens')
    args = parser.parse_args()

    colonne_liens_id = get_table()
    # On extrait les liens correspondants à chaque site
    if args.site == "nadine":
        liste_liens = get_liens(colonne_liens_id, debut = 0, fin = 76)
    elif args.site == "jackie":
        liste_liens = get_liens(colonne_liens_id, debut = 76, fin = 149)
    elif args.site == "mercotte":
        liste_liens = get_liens(colonne_liens_id, debut = 149, fin = 199)
    elif args.site == "marmiton":
        liste_liens = get_liens(colonne_liens_id, debut = 199, fin = 349)
    elif args.site == "elle":
        liste_liens = get_liens(colonne_liens_id, debut = 349, fin = 500)
    get_texte(liste_liens, args.site)

if __name__ == "__main__":
    main()
```

nettoyage.py :

Une fois l'ensemble de nos contenus textuels obtenus, nous avons pu commencer la rédaction d'un programme permettant d'isoler les portions pertinentes de chaque page. Pour cela, nous avons pris la décision d'utiliser des expressions régulières plutôt que ***BeautifulSoup*** à cause des structures HTML beaucoup trop variées au sein parfois d'un même site. Le script ***nettoyage.py*** contient ainsi 5 fonctions permettant de récupérer le corps de chaque recette (liste des ingrédients et instructions pour chaque site.

```
def nettoyage_jackie(liste_fichiers):

   for fichier in liste_fichiers:
       with open(fichier, "r", encoding="utf8") as file:
           texte_full = file.read()
           texte = re.search("Ingrédients(.*?)(?=Les participants|La petite
(H|h)istoire|(A|a)vec cette recette|Impression)", texte_full, re.DOTALL)
           texte_exception(fichier, texte)
```

Les fonctions de nettoyage pour les sites de Jackie, Marmiton et Elle ne requièrent que très peu de lignes de codes là ou celles de Mercotte et Nadine requièrent la gestion de nombreuses exceptions ou variations de structure :

```
def nettoyage_mercotte(liste_fichiers):

   for fichier in liste_fichiers:
       with open(fichier, "r", encoding="utf8") as file:
           texte_full = file.read()
           try:
               if fichier == Path("dumps-text/mercotte/146_mercotte_dump.txt"):
                   texte = re.search("La recette(.*?)(?=Imprimer la (R|r)ecette)", texte_full, re.DOTALL)
                   texte_exception(fichier, texte)
               elif fichier == Path("dumps-text/mercotte/195_mercotte_dump.txt"):
                   texte = re.search("Pour 6 personnes : préparation 10 min, cuisson 40min(.*?)(?=Imprimer la
(R|r)ecette)", texte_full, re.DOTALL)
                   texte_exception(fichier, texte)
               else:
                   texte = re.search("La recette(.*?)(?=Explications? utiles? ou futiles?|Imprimer la
(R|r)ecette|Langoustines\? Saint Jacques\?)", texte_full, re.DOTALL)
                   texte_exception(fichier, texte)
           except:
               if fichier == Path("dumps-text/mercotte/159_mercotte_dump.txt"):
                   texte = re.search("La recette :(.*?)(?=On peut présenter dans des tasses avec possibilité de
se servir ou non de noisettes)", texte_full, re.DOTALL)
                   texte_exception(fichier, texte)
               else:
                   texte = re.search("Version rapide :(.*?)(?=Imprimer la Recette)", texte_full, re.DOTALL)
                   texte_exception(fichier, texte)

           # On aime BEAUCOUP moins Mercotte ! COLÈRE !
```

Dépendamment de la date de publication, certaines recettes commençaient de manière différente à la majorité du corpus ou bien contenaient des portions supplémentaires dédiées à l'histoire d'un ingrédient, d'une recette ou bien d'une célébration. L'inclusion des ***Try Except*** nous a initialement permis d'identifier les fichiers ne répondant pas à l'expression régulière initiale puis d'adapter ces dernières en conséquence.

Ces contenus récupérés étaient ensuite enfin tous traités afin de supprimer d'éventuels espaces ou tabulations puis sauvegardés dans des fichiers puis dossiers nommés de manière similaire à ceux introduits précédemment. L'ensemble de ces fichiers était contenu dans le dossier ***dumps-traites.***

```python
def texte_exception(fichier, texte):

    chemin = f"dumps-traites/{fichier.parent.name}/{fichier.name}"
    print(chemin)
    print(texte)
    with open(chemin, "w", encoding="utf8") as file_traite:
        texte_trop_beau = re.sub(r"\s+", " ", texte.group(0).strip())
        file_traite.write(texte_trop_beau)
```

Afin d'illustrer les deux programmes présentés, voici un extrait du contenu textuel d'une page du site de Jackie avant et après nettoyage.

Avant nettoyage :

Ces contenus textuels contiennent, en plus des instructions et de la liste des ingrédients, de nombreuses indications ainsi que des commentaires utilisateurs et autres onglets ou sections du site.

Après nettoyage :

```
Ingrédients : Pour 12 cristallines • 1 ou 2 poires selon la quantité que vous voulez faire • 200 g de sucre blanc • 20 cl d'eau • le jus 1/2 citron
Préparation des cristallines de poires Préparation : 15 mn Cuisson : 1h30 four 100° .
• Préparez un sirop et • Faites bouillir l'eau, le sucre et le jus de citron dans une casserole
• Lavez et essuyez les poires. • Ne les épluchez pas. • A l'aide la mandoline (attention les doigts!),
coupez les en tranches de 1 à 2 mn d'épaisseur, en gardant la peau et la queue.
• Préchauffez votre four à 100° • Plongez les tranches de poires une à une, attention de ne pas les faire se chevaucher.
• Laissez pocher 1 à 2 mn.• • Les poires doivent être translucides.
• Déposez les poires sur une plaque Silpat ou du papier sulfurisé.
• Mettez ensuite votre plaque au four préchauffée pendant 1h30 environ.
• C'est suffisant, certaines recettes vous parlent de 6 h je trouve que c'est tout à fait inutile.
• Laissez ensuite refroidi à température ambiante. • A conserver dans un contenant en verre.
• Idéal pour décorer vos pâtisseries ou panna cotta ou mouse de fruits .
```

creation_tableau.py :

Une fois toutes nos recettes nettoyées et stockées, nous avons pu faire en sorte de repasser à un format facilement exploitable pour l'entraînement de nos modèles. Afin de recréer un fichier sous format tabulaire, nous avons commencé par extraire l'ensemble des colonnes de notre tableau initial ainsi que le contenu de chacun de nos fichiers :

```
def get_table():
    tableau = pd.read_csv("liens.csv")
    colonnes = tableau[["id", "liens", "vegetarien", "vegan", "crudivore", "sans_gluten",
"crudivore", "sans_gluten", "alixproof", "sans_noix", "sucré/salé"]]
    colonnes = tableau[tableau["id"] > 0] # On enlève les lignes avec les totaux
    colonnes["id"] = colonnes["id"].astype(int) # On convertit tout en entier
    colonnes["vegetarien"] = colonnes["vegetarien"].astype(int)
    colonnes["vegan"] = colonnes["vegan"].astype(int)
    colonnes["crudivore"] = colonnes["crudivore"].astype(int)
    colonnes["sans_gluten"] = colonnes["sans_gluten"].astype(int)
    colonnes["alixproof"] = colonnes["alixproof"].astype(int)
    colonnes["sans_noix"] = colonnes["sans_noix"].astype(int)

    return colonnes

get_table()
def get_textes(dossier):
    liste_textes = []
    listes_fichiers = sorted(dossier.glob("*/*.txt"), key=lambda fichier:
int(re.match(r'(\d+)', fichier.name).group(0)))

    for fichier in listes_fichiers:
        with open(fichier, "r", encoding="utf8") as f:
            texte = f.read()
            liste_textes.append(texte)

    return liste_textes
```

Nous créons également une liste de nos recettes lemmatisées grâce à la fonction suivante :

```
def lemmatisation(liste_textes):
    import spacy
    nlp = spacy.load("fr_core_news_md")
    liste_textes_lemma = []
    for texte in liste_textes:
        doc = nlp(texte)
        liste_texte_lemma = []
        for mot in doc:
            lemme = mot.lemma_
```

```
            liste_texte_lemma.append(lemme)
        texte_lemma = " ".join(liste_texte_lemma)
        liste_textes_lemma.append(texte_lemma)

    return liste_textes_lemma
```

Nous pouvons enfin combiner ces textes à notre tableau initial et obtenir un fichier final utilisé pour l'entraînement de nos modèles.

```
def insertion_textes(liste_textes, colonnes, liste_textes_lemma):
    liens = colonnes.columns.get_loc("liens")

    avant = colonnes.iloc[:, : liens+1]
    après = colonnes.iloc[:, liens+1 :]
    avant["textes"] = liste_textes
    avant["textes_lemma"] = liste_textes_lemma
    tableau = pd.concat([avant, après], axis=1)
    print(tableau)
    tableau.to_csv("recettes.csv", index=False, encoding='utf-8-sig')
```

Le fichier résultant de ces trois programmes d'extraction et de nettoyage se nomme ***recettes.csv*** et se présente de la manière suivante :

	A	B	C	D	E	F	G	H	I	J	K
		liens	textes	textes_lemma	vegetarien	vegan	crudivore	sans_gluten	alixproof	sans_noix	sucré/salé
1		https://quandnadcuisine.fr/salade-de-quinoa-a-la-mexicaine/	Let's go en cuisine pour préparer cette salade de quinoa à la mexicaine! Imprimer Épingler la recette 5 de 1 vote Salade de quinoa à la mexicaine Temps de préparation15 minutes min Temps de cuisson15 minutes min Type de plat: Plat principalCuisine: Mexicaine Mots-clés: avocat, gâteaux maison, haricots rouges, poivrons rouges, quinoa, salade Portions: 4 Ingrédients 200 g de quinoa1 petit oignon rouge (ou 1/2 s'il est gros)1 poivron rouge250 g de haricots rouges250 g de maïs1 avocatPour la sauce:1/2 cc de moutarde25 g de vinaigre de cidre20 g de miel25 g d'huile d'olive30 g de jus de citron vert10 g de coriandre ciselée1 gousse d'ailquelques gouttes de tabascosel, poivre InstructionsCommencer par rincer le quinoa sous l'eau froide puis le faire cuire dans une casserole d'eau bouillante salée le temps indiqué sur le paquet.Passer le quinoa cuit sous l'eau froide puis égoutter.Le placer dans un saladier avec l'oignon émincé, le poivron en dés, les haricots rouges et le maïs égouttés ainsi que l'avocat en dés.Mélanger tous les ingrédients de la sauce (ciseler la coriandre et écraser l'ail avant de les ajouter).Verser dans le saladier et mélanger.Servir la salade bien fraîche.	Let' s go en cuisine pour préparer ce salade de quinoa à le mexicain ! Imprimer épingler le recette 5 de 1 vote salade de quinoa à le mexicain temps de préparation15 minute min temps de cuisson15 minute min type de plat : plat principalcuisine : mexicain mot-clé : avocat , gâteau maison , haricot rouge , poivron rouge , quinoa , salade portion : 4 ingrédient 200 gramme de quinoa1 petit oignon rouge (ou 1/2 se il être gros)1 poivron rouge250 gramme de haricot rouges250 gramme de maïs1 avocatpour le sauce:1/2 cc de moutarde25 gramme de vinaigre de cidre20 gramme de miel25 gramme de huile de olive30 gramme de jus de citron vert10 gramme de coriandre ciselée1 gouss de ailquelque goutte de tabascosel , poivre instructionscommencer par rincer le quinoa sou le eau froid pouvoir le faire cuire dans un casserole de eau bouillant saler le temps indiquer sur le paquet . passer le quinoa cuit sou le eau froid pouvoir égoutter . le placer dans un saladier avec le oignon émincer , le poivron en dé , le haricot rouge et le maïs égoutter ainsi que le avocat en dé . mélanger tout le ingrédient de le sauce (ciseler le coriandre et écraser le ail avant de le ajouter).verser dans le saladier et mélanger . servir le salade bien frais .	1	1	0	1	1	1	salé
2		https://quandnadcuisine.fr/bricks-au-thon-et-tomates-sechees/	Let's go en cuisine pour préparer ces bricks au thon et tomates séchées! Imprimer Épingler la recette 5 de 1 vote Bricks au thon et au tomates séchées Temps de préparation25 minutes min Temps de cuisson15 minutes min Type de plat: Apéritif, Brunch, Entrée, Plat principal Mots-clés: feuilles de brick, fromage frais, mozzarella, thon, tomates séchées Portions: 16 Ingrédients 240 g de thon (poids net égoutté)1 petit oignon rouge6 tomates séchées à l'huile1 oeuf1 cs de fromage frais (type St Môret)2 cc de mélange d'épices (ail, basilic, coriandre, curcuma et cumin)sel, poivre8 feuilles de brickenviron 60 g de mozzarella râpée InstructionsDans un petit saladier, émietter le thon puis ajouter l'oignon finement haché, les tomates séchées en petits morceaux, l'oeuf, le fromage frais et les épices.Saler et poivrer puis bien mélanger.Préchauffer le four à 200° (chaleur tournante).Couper les feuilles de brick en deux avec des ciseaux.Replier la moitié de feuille en deux de façon à obtenir une bande. Déposer une cuillère de garniture en bas, ajouter une belle pincée de mozzarella et plier comme sur le schéma.Déposer les bricks au fur et à mesure sur une plaque tapissée d'une feuille de cuisson.Avec un pinceau, les badigeonner avec un peu d'huile du bocal de tomates séchées.Enfournez et laisser cuire environ 15 min (jusqu'à ce que les bricks soient dorées).Notes:A déguster, en apéro, en entrée ou pourquoi pas en plat principal pour le soir, avec une bonne salade.	Let' s go en cuisine pour préparer ce brick au thon et tomate sécher ! Imprimer épingler le recette 5 de 1 vote Bricks au thon et au tomate sécher temps de préparation25 minute min temps de cuisson15 minute min type de plat : apéritif , brunch , entrée , plat principal mot-clé : feuille de brick , fromage frais , mozzarella , thon , tomate sécher portion : 16 ingrédient 240 gramme de thon (poids net égoutté)1 petit oignon rouge6 tomate sécher à le huile1 oeuf1 c de fromage frais (type st môret)2 cc de mélange de épice (ail , basilic , coriandre , curcuma et cumin)sel , poivre8 feuille de brickenviron 60 gramme de mozzarella râpée InstructionsDans un petit saladier , émietter le thon pouvoir ajouter le oignon finement hacher , le tomate sécher en petit morceau , le oeuf , le fromage frais et le épice . saler et poivrer pouvoir bien mélanger . préchauffer le four à 200 degré (chaleur tournante).couper le feuille de brick en deux avec un ciseau . replier le moitié de feuille en deux de façon à obtenir un bande . déposer un cuillère de garniture en bas , ajouter un bel pincée de mozzarella et plier comme sur le schéma . déposer le brick au fur et à mesure sur un plaque tapisser de un feuille de cuisson . avec un pinceau , le badigeonner avec un peu de huile de bocal de tomate sécher . enfournez et laisser cuire environ 15 min (jusque à ce que le brick être dorées).noter : avoir déguster , en apéro , en entrée ou pourquoi pas en plat principal pour le soir , avec un bon salade .	0	0	0	0	0	1	salé

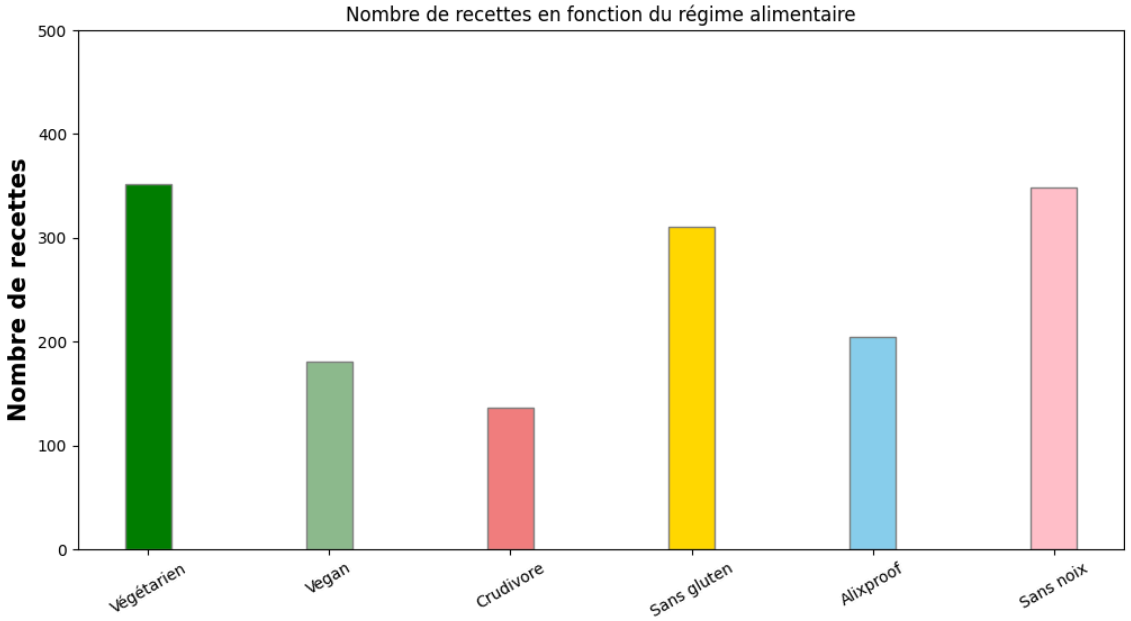
Nous tenons également à ajouter que nous avons fait en sorte d'exclure les indices évidemment de chaque recette. Nous avons par exemple retiré tous les tags mentionnant des régimes tels que "végétarien" ou "pescetarien".

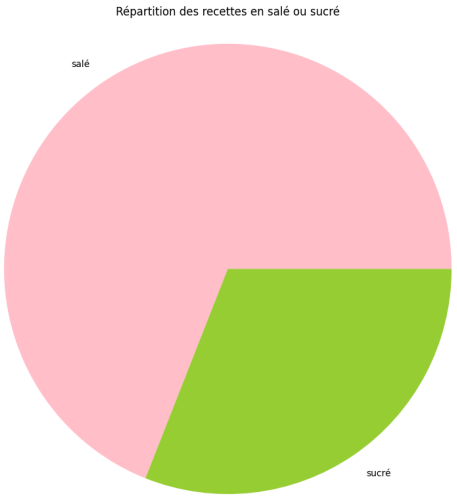
À partir du corpus maintenant complet, nous pouvons établir un bilan des données obtenues.

1.4. Bilan des données récoltées

Notre corpus comporte 500 recettes. Pour avoir plus d'informations sur notre corpus nous avons décidé de créer un script python ***statistiques.py*** dans lequel nous essayons de faire des statistiques afin d'explorer nos données. Avec notre tableau de données sur notre corpus nous avons rempli une dataclass ***recettes*** qui nous à permis d'effectuer différents calculs. Au sujet de nos données maintenant, les recettes contiennent en moyenne 231.694 mots.

	végétarien	végan	crudivore	Sans gluten	Alix Proof	Sans noix	salé	sucré
Chiffres	352	181	136	311	205	348	345	155
Pourcentage	70.4	36.2	27.2	62.2	41.0	69.6	69.0	31.0





Nous pouvons observer que certains régimes, malgré notre extraction manuelle, sont sous-représentés. C'est notamment le cas des régimes **alixproof, vegan et crudivore**, ce dernier contenant deux fois moins de recette que le régime végétarien. Ces chiffres s'expliquent par le fait que la majeure partie des recettes sucrées soient végétariennes et que l'ensemble des recettes végan le sont également. Les recettes n'impliquant aucune cuisson ou ne contenant aucun ingrédient exclu du régime alixproof se font elles extrêmement rares indépendamment de la source utilisée.
Le corpus comporte une majorité de recettes salées soit près de 69% de salé contre 31% de sucré. Ce paramètre peut éventuellement influencer les pourcentages de recettes pour chaque régime. Par exemple, toutes les recettes sucrées sont végétariennes alors que pour la plupart des autres régimes ce n'est pas un élément discriminant.

2. Entraînement des modèles

Dans le cas de notre classification binaire, 1 représente les recettes qui sont compatibles avec les régime alimentaire testé et 0 représente la classe des recettes incompatibles avec le régime.

Nous allons essayer d'entraîner nos trois modèles sur chacun de nos 6 régimes alimentaires.
Avant de choisir nos modèles nous avons fait des tests sur notre corpus en utilisant Naive Bayes. Les tests n'étant pas concluants nous nous sommes tournés vers 3 autres classifieurs.

Pour l'entraînement, nous avons donc choisi d'utiliser SVM, KNN et le DTC.

- En premier l'algorithme des **Support Vector Machines** (SVM). Les SVM vont nous aider à séparer en deux catégories nos recettes en prenant l'hyperplan qui sépare au mieux nos données.
- Dans un second temps, nous avons utilisé le classifieur des **K plus proches voisins** (K Nearest Neighbours). Ce dernier utilise des mesures variées, comme les distances euclidiennes ou de Manhattan, et utilise ces résultats comme élément de poids lors du choix de la classe d'un document.
- Pour le troisième classifieur nous avons choisi le modèle **Decision Tree Classifier.** Cet algorithme commence par choisir les meilleurs attributs par classe et construit des arbres de décisions de manière récursive en répétant ces étapes: trouver le meilleur attribut discriminant et création d'une nouvelle branche. Il continue jusqu'à ce qu'une de ces trois conditions soient remplies: les éléments restants appartiennent tous à la même classe, il n'y à plus d'attribut discriminant, il n'y à plus d'éléments à classer.

Pour réaliser les tests nous avons utilisé la librairie ***scikitlearn***. Nous avons créé un script python par modèle d'entraînement, la structure de ces scripts python est la même pour les trois.
Dans ce script nous avons choisi d'enlever les ***stop words*** sauf "pas". En effet avoir le mot "lait" associé a "pas" peut donner des indications importantes aux modèles.
Pour lancer le script, il faut lui donner un argument. Un parser d'argument est utilisé pour choisir le régime que nous voulons tester:

```
parser = argparse.ArgumentParser(description="Choix du régime alimentaire pour l'entraînement")
parser.add_argument('regime', choices=['vegetarien', 'vegan', 'crudivore', 'sans_gluten',
'alixproof', 'sans_noix', "sucré/salé"], help= "Choix du régime alimentaire")
```

On commence ensuite par récupérer les colonnes qui nous intéressent dans le tableau, soit les colonnes ***textes, textes_lemma*** et celle correspondant au régime souhaité en argument.
Pour les données d'entraînement, nous avons essayé avec la colonne ***textes*** uniquement puis la colonnes ***textes_lemma*** et après réflexion nous avons mixé les deux colonnes en un seul corpus de train.

```
corpus_dataframe = pd.read_csv('recettes.csv', header=0, usecols=[2, 3, index], names=['textes', 'textes_lemma', regime])
corpus_dataframe = corpus_dataframe.dropna(subset=['textes'])

train_corpus, test_corpus=train_test_split(corpus_dataframe, test_size=0.25, random_state=42,stratify=corpus_dataframe[regime])

train_text = train_corpus['textes'] + ' ' + train_corpus['textes_lemma']
test_text = test_corpus['textes'] + ' ' + test_corpus['textes_lemma']
```

Le corpus obtenu est ensuite divisé en un corpus de train et un corpus de tests. Nous avons choisi de diviser ***75% de train et 25% de tests.***

Pour la vectorisation du corpus nous avons utilisé **TF IDF**. La technique de vectorisation TF IDF n'est peut-être pas la plus optimale mais est la plus simple à notre connaissance. Cet outil de vectorisation peut éventuellement être remplacé par la technique word2vec en utilisant doc2vec comme outil de vectorisation des recettes.

Ces scripts nous permettent également de générer des matrices de confusions pour chaque test et des tableaux contenant les métriques de tests suivantes: précision, rappel et f-mesure pour nos deux classes puis une mesure d'accuracy et enfin les macro et micro précision, rappel et f-mesure.

Pour chaque régime nous allons maintenant vous présenter les résultats des trois modèles d'entraînement.

Nos hypothèses sont les suivantes:

- ***Pour des régimes dont l'équilibre entre les classes n'est pas respecté, les modèles risquent de moins bien repérer la classe minoritaire. Par exemple, le régime végétarien est surreprésenté alors que le régime crudivore est sous représenté.***
- ***En dehors de l'équilibre des classes, les attributs discriminants et leur clarté risquent d'influencer grandement le succès des modèles de classification.***
  Par exemple pour ces régimes:

Régime	**Végétarien**	**Alixproof**	**Sans-gluten**
Attributs discriminants	**Viande** : bœuf, porc, jambon, agneau, saucisse, poulet… **Poisson** : saumon, colin, crevettes…	**Produits laitiers :** lait (de vache, de jument et de chèvre), crème, fromages, beurre… Fruits de mer, pommes de terre, patates douces, oeufs…	**Céréales :** blé, seigle, orge, épeautre **Formes :** farine, boulgour, semoule, seitan, sauce soja, extrait de levure, vinaigre de malt, bière, friture (Panco)

On peut voir que pour le régime végétarien les attributs sont assez clairs alors que pour les régimes alixproof et sans gluten, les allergènes prennent des formes multiples et des noms différents.

Nous faisons donc les hypothèses suivantes :

- ***Le régime végan sera l'un des mieux reconnus de par son équilibre des classes et une liste d'attribut clair et suffisamment restreinte.***
- ***Le régime crudivore sera lui le moins bien reconnu de par son déséquilibre de classe et ses attributs reposant sur des verbes, des noms d'ustensiles et de techniques variant d'un cuisinier à un autre.***

Nous avons développé la comparaison de chaque classifieur pour chaque régime. Si vous souhaitez directement obtenir une analyse comparative des régimes et des modèles, nous vous invitons à passer à la partie conclusion.

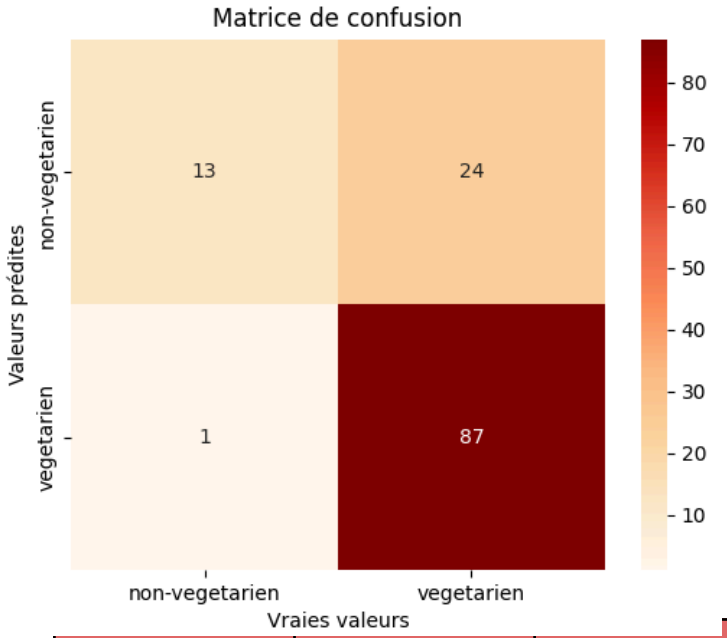
5.1. Végétarien

Le régime végétarien, consiste à ne pas consommer de viande animale comme le saucisson et les poissons. Ce régime peut être suivi pour de nombreuses raisons, elles peuvent être religieuses, médicales ou par convictions politiques.

Pour ces premiers tests, on va prendre comme mesure de comparaison la macro f-mesure pour tester l'efficacité du modèle. La classe végétarienne étant majoritaire et nous voulons voir si le modèle reconnaît bien les recettes non végétarienne.

a) SVC

Voici la matrice de confusion et le tableau présentant les mesures de tests pour les SVM :

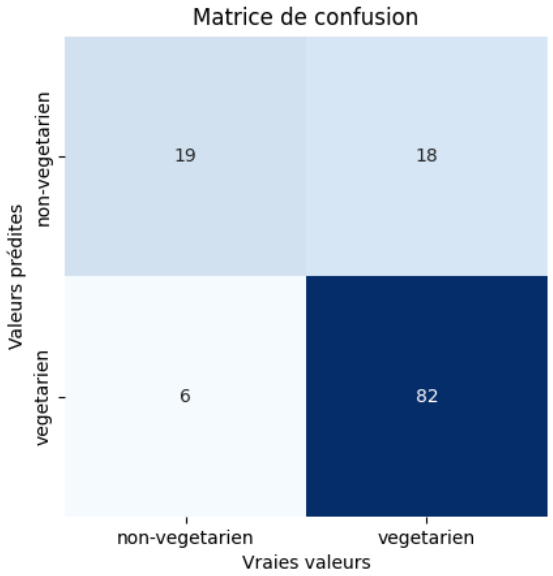


Pour ce premier modèle, la macro f-mesure est de 0.69. On observe un écart important entre précision et rappel. La mesure de rappel pour les recettes non-végétarienne est assez basse à 0.35 ce qui n'est pas surprenant pour la classe minoritaire.

	Précison	Rappel	F-mesure	Support
0	0.93	0.35	0.51	37
1	0.78	0.99	0.87	88
Accuracy			0.80	125
Macro AVG	0.86	0.67	0.69	125
Micro AVG	0.83	0.80	0.77	125

b) KNN

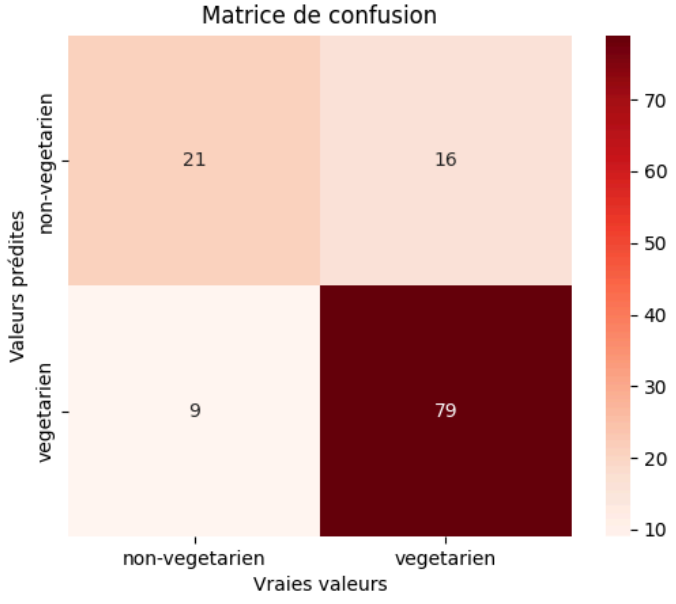
Voici la matrice de confusion et le tableau présentant les mesures de tests pour le modèle KNN:



Tout comme pour le classifieur SVC, il semblerait que la classe la mieux reconnue soit celle des recettes végétariennes. Nous observons cependant une légère augmentation des valeurs de macro-moyenne et de précision et rappel pour nos deux classes. Cette augmentation est d'autant plus importante pour la classe des recettes non végétariennes dont la f-mesure a augmenté de 0.10 points. Si la valeur de précision a baissé de plus de 15 points, l'écart entre la précision et le rappel s'est lui réduit passant de 0.6 à 0.25 points. Cet écart réduit ainsi que meilleures valeurs de moyenne indique que le classifieur KNN est plus performant que le SVC pour la reconnaissance des recettes végétariennes.

	Précison	Rappel	F-mesure	Support
0	0.76	0.51	0.61	37
1	0.82	0.93	0.87	88
Accuracy			0.81	125
Macro AVG	0.79	0.72	0.74	125
Micro AVG	0.80	0.81	0.80	125

c) DTC



Voici la matrice de confusion et le tableau présentant les mesures de tests pour le modèle DTC:

Les valeurs obtenues à l'aide du classifieur DTC sont extrêmement similaires à celles obtenues à l'aide du classifieur KNN. Les macro et micro-moyennes sont par exemple équivalentes. Les seules différences majeures pouvant être observées se trouvent au niveau de la précision de la classe non végétarienne plus élevée de 0.06 points dans le cas du KNN malgré une macro-précision plus élevée pour le DTC.

	Précison	Rappel	F-mesure	Support
0	0.70	0.57	0.63	37
1	0.83	0.90	0.86	88
Accuracy			0.80	125
Macro AVG	0.77	0.73	0.75	125
Micro AVG	0.79	0.80	0.79	125

d) Conclusion

Voici un tableau récapitulatif des macro f-mesure des trois modèles:

	SVC	KNN	DTC
Macro-AVG	0.69	0.74	0.75

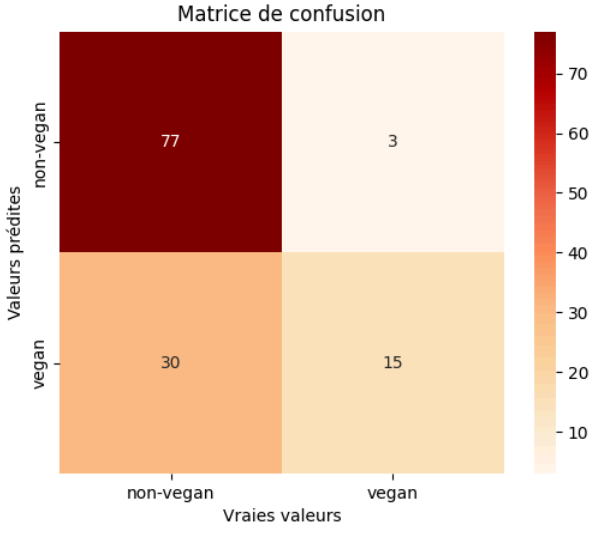
On peut dire que pour ce premier régime, le modèle entraîné avec les SVM est moins efficace que Knn et le DTC. Pour ce qui est des deux derniers, ils sont équivalents, il faudra affiner les hyperparamètres ou inclure des features discriminantes afin d'améliorer les résultats et pouvoir prendre une décision.

5.2. Végan

Plus restrictif que le régime végétarien, le régime végan exclut tous les produits d'origine animale ce qui inclut en plus de la viande, les œufs, le lait. La consommation de produits comme le miel, la cire d'abeille ou le cuir sont souvent exclues également. Comme le régime végétarien, le véganisme est souvent adopté par convictions spirituelles, politiques ou médicales.
Pour ces deuxième tests, on va prendre comme mesure de comparaison la macro f-mesure pour tester l'efficacité du modèle. La classe vegan étant minoritaire et nous voulons voir si le modèle reconnaît bien ces recettes.

a) SVC

Voici la matrice de confusion et le tableau présentant les mesures de tests pour les SVM :

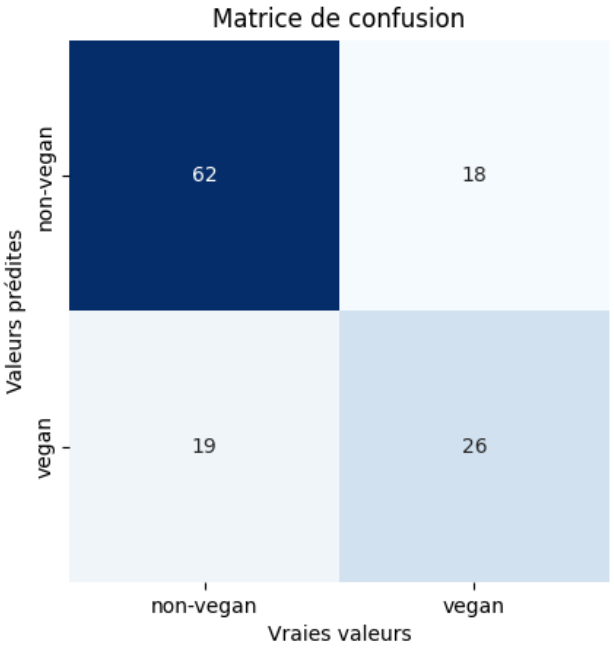


La mesure de précision est basse pour la classe vegan cette fois-ci. Ce qui veut dire qu'il y a beaucoup de recettes vegan que le modèle ne reconnaît pas comme vegan, la classe vegan étant minoritaire (0.33). Pour ce qui est du modèle en lui-même, la macro f-mesure est de 0.65.

	Précison	Rappel	F-mesure	Support
0	0.72	0.96	0.82	80
1	0.83	0.33	0.48	45
Accuracy			0.74	125
Macro AVG	0.78	0.65	0.65	125
Micro AVG	0.76	0.74	0.70	125

b) KNN

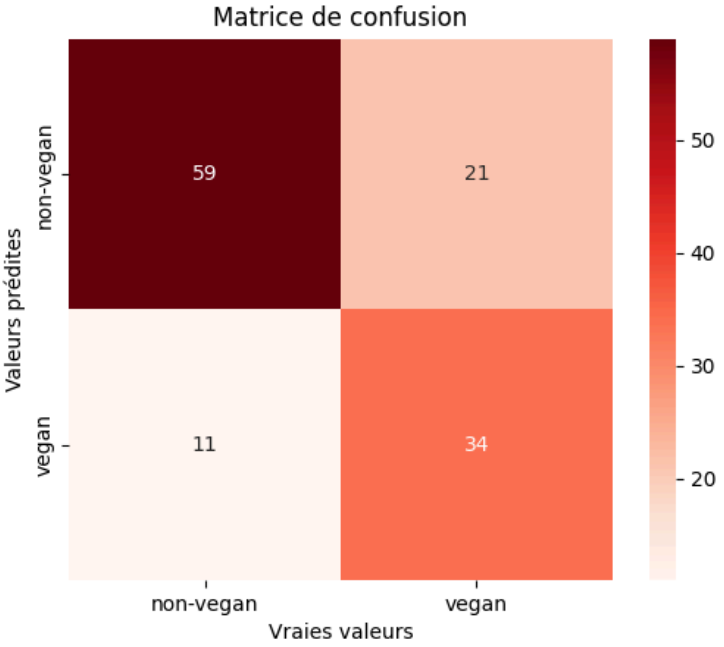
Voici la matrice de confusion et le tableau présentant les mesures de tests pour le modèle KNN:



Les résultats obtenus à l'aide du classifieur KNN semblent légèrement meilleurs que ceux obtenus à l'aide du SVC malgré des valeurs parfois équivalentes contrairement aux différences majeurs observables lors de l'entraînement sur le régime végétarien. Malgré des mesures de moyennes similaires (0.68 contre 0.65 et 0.70 contre 0.70), KNN permet ici d'obtenir un équilibre quasi parfait des mesures de rappel et de précision pour chaque classe, tout l'inverse du SVC.

	Précison	Rappel	F-mesure	Support
0	0.77	0.78	0.77	80
1	0.59	0.58	0.58	45
Accuracy			0.70	125
Macro AVG	0.68	0.68	0.68	125
Micro AVG	0.70	0.70	0.70	125

c) DTC



Voici la matrice de confusion et le tableau présentant les mesures de tests pour le modèle DTC:

L'écart entre précision et rappel est plus important que celui du modèle KNN mais le reste des mesures sont plus élevées. La classe minoritaire des recettes vegan est mieux reconnue que dans les deux autres modèles.

	Précison	Rappel	F-mesure	Support
0	0.84	0.74	0.79	80
1	0.62	0.76	0.68	45
Accuracy			0.74	125
Macro AVG	0.73	0.75	0.73	125
Micro AVG	0.76	0.74	0.75	125

d) Conclusion

Voici un tableau récapitulatif des macro f-mesure des trois modèles:

	SVC	KNN	DTC
Macro-AVG	0.65	0.68	0.73

En conclusion, les classifieurs KNN et DTC sont plus efficaces que le SVC. Leurs valeurs de précision, rappel et moyennes sont généralement plus hautes. Le modèle qui prédit le mieux si une recette est végan ou non est le DTC.

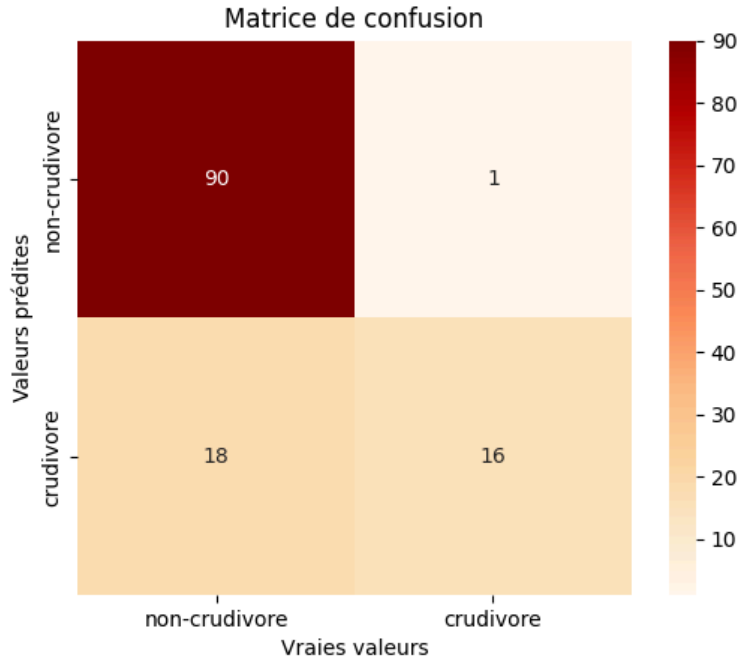
5.3. Crudivore

Le régime crudivore[4] consiste à ne manger uniquement des produits crus. Les produits lacto-fermentés par exemple ne sont pas cuits. Toutes les techniques de transformations des aliments utilisant la chaleur sont prohibés comme les produits fumés ou cuits à la vapeur.

Tout comme pour les régimes vegan et végétariens, nous faisons ici face à une situation de déséquilibre de classe dans laquelle les recettes crudivores incarnent la classe sous-représentée. Nous allons donc majoritairement juger la performance de chaque modèle en fonction des valeurs de macro-moyenne.

a) SVC

Voici la matrice de confusion et le tableau présentant les mesures de tests pour les SVM:



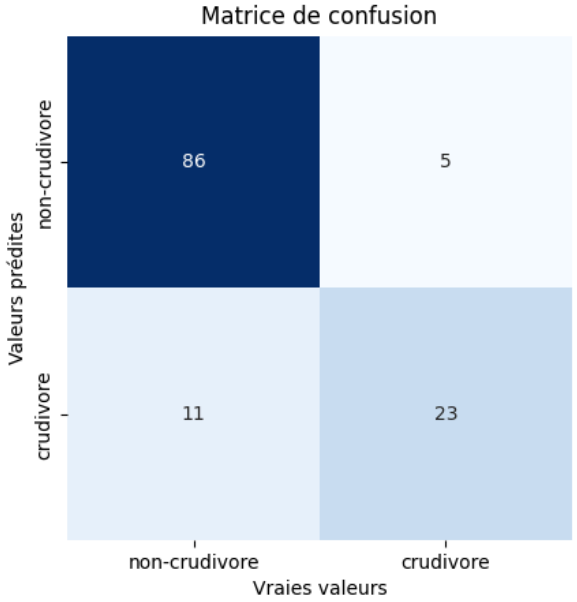
Les résultats obtenus semblent indiquer que le modèle SVC est très performant pour la reconnaissance des recettes non crudivore. Il l'est cependant beaucoup moins pour les recettes crudivores incarnant ici notre classe minoritaire, leurs deux f-mesures étant respectivement 0.90 et 0.63. Nous observons de nouveau un écart majeur entre la précision et le rappel de la classe crudivore avec une précision quasiment deux fois plus élevée.

	Précison	Rappel	F-mesure	Support
0	0.83	0.99	0.90	91
1	0.94	0.47	0.63	34
Accuracy			0.85	125
Macro AVG	0.89	0.73	0.77	125
Micro AVG	0.86	0.85	0.83	125

[4] https://mariesophiel.com/chef-cuisine-crue/

b) KNN

Voici la matrice de confusion et le tableau présentant les mesures de tests pour le modèle KNN:

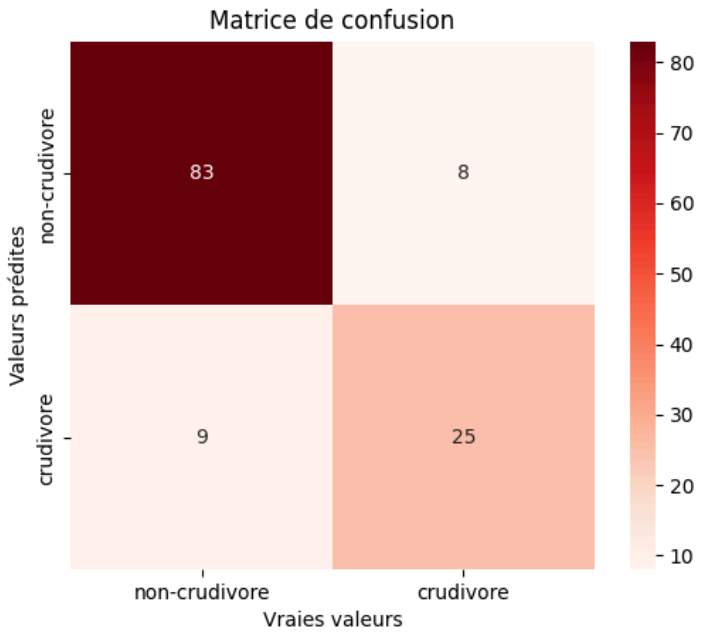


Les résultats obtenus sont plus satisfaisants que pour le SVC. Nous gagnons notamment 0.10 points de macro-moyenne pour la classe crudivore et 0.20 points pour sa valeur de rappel. Sa valeur de précision est elle légèrement plus basse que pour SVC mais l'écart entre cette dernière et le rappel étant réduit, nous pouvons admettre que les valeurs obtenues avec KNN témoignent d'un modèle plus performant et adapté à notre tâche de classification binaire.

	Précison	Rappel	F-mesure	Support
0	0.89	0.95	0.91	91
1	0.82	0.68	0.74	34
Accuracy			0.87	125
Macro AVG	0.85	0.81	0.83	125
Micro AVG	0.87	0.87	0.87	125

c) DTC

Voici la matrice de confusion et le tableau présentant les mesures de tests pour le modèle DTC:



Comme pour les autres modèles, la classe minoritaire des recettes crues est moins bien reconnue que celle des recettes cuites. On peut voir que l'écart entre précision et rappel est moins important ici que pour le SVM et le KNN.
En revanche, la macro f-mesure est la même que pour KNN.

	Précison	Rappel	F-mesure	Support
0	0.90	0.91	0.91	91
1	0.76	0.74	0.75	34
Accuracy			0.86	125
Macro AVG	0.83	0.82	0.83	125
Micro AVG	0.86	0.86	0.86	125

d) Conclusion

Voici un tableau récapitulatif des macro f-mesure des trois modèles:

	SVC	KNN	DTC
Macro-AVG	0.77	0.83	0.83

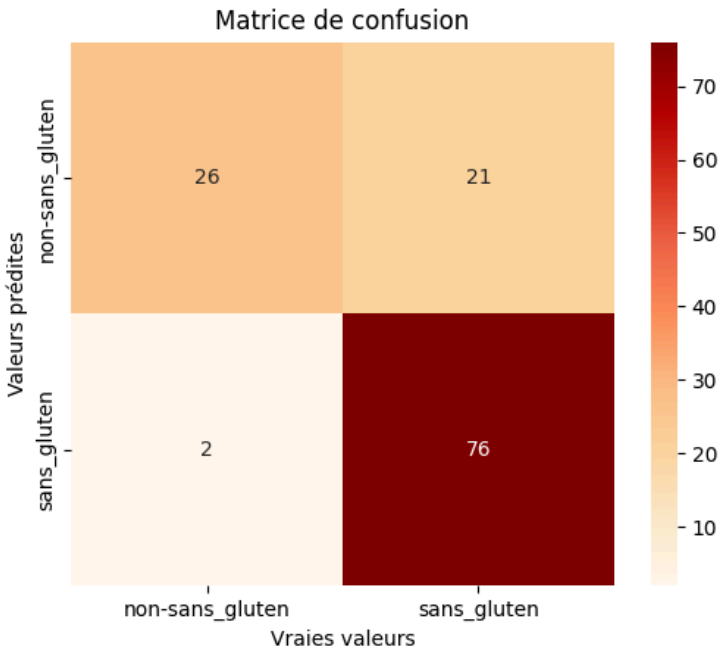
La moyenne macro est moins bonne pour les trois tests que la micro ce qui est normal car les modèles reconnaissent mieux la classe majoritaire qui est la classe des recettes non-crudivores.
Les deux modèles qui reconnaissent le mieux le régime crudivore sont le KNN et le DTC. Pour les départager on peut voir que le DTC à un meilleur équilibre de rappel et de précision mais que KNN à des valeurs de précision et rappels plus élevés et une accuracy plus importante.

5.4. Sans Gluten

Le régime sans gluten consiste à retirer tous les aliments contenant du gluten. Le gluten est la protéine contenu dans certaines céréales. Le blé, le seigle, l'orge et l'épeautre en contiennent. Il peut se présenter sous différentes formes: farine, boulgour, semoule, seitan, sauce soja, extrait de levure, vinaigre de malt, bière, friture (Panco) ... Ce régime peut être indiqué comme traitement pour des maladies auto-immunes comme la maladie coeliaque ou des problèmes digestifs comme l'IBS ou le SIBO.
Le nombre de recettes correspondant au régime sans-gluten reste similaire à celui obtenu pour les recettes végétariennes. Nous allons ainsi prendre en compte les valeurs de macro-moyenne afin de donner autant d'importance à notre classe minoritaire qu'à notre classe majoritaire.

a) SVC

Voici la matrice de confusion et le tableau présentant les mesures de tests pour les SVM:



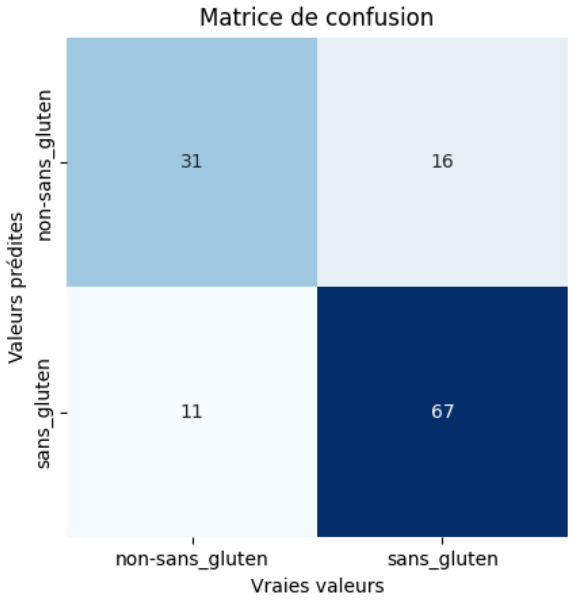
Les résultats obtenus indiquent que la classe sans gluten est ici mieux reconnue que la classe minoritaire (différence de 0.20 points de f-mesure). De plus, la classe sans gluten dispose d'un rappel plus élevé que la précision et la tendance inverse s'observe pour la seconde classe. La différence entre les moyenne de chaque classe n'étant pas extrêmement élevée, nos valeurs de micro et de macro restent similaires (0.78 contre 0.80)

.

	Précison	Rappel	F-mesure	Support
0	0.93	0.55	0.69	47
1	0.78	0.97	0.87	78
Accuracy			0.82	125
Macro AVG	0.86	0.76	0.78	125
Micro AVG	0.84	0.82	0.80	125

b) KNN

Voici la matrice de confusion et le tableau présentant les mesures de tests pour le modèle KNN:



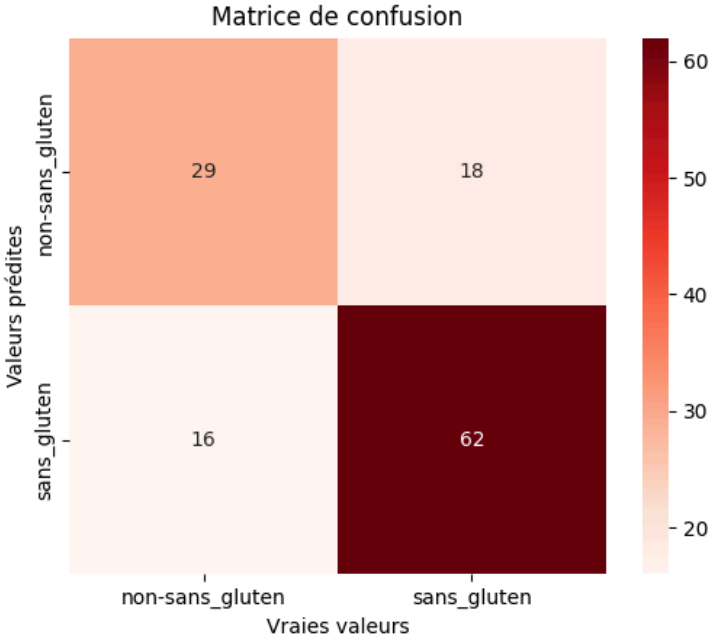
Les résultats obtenus avec le classifieur KNN sont généralement plus bas que ceux obtenus avec SVC. Les macro moyenne, précision, rappel, micro moyenne, précision et rappels sont en moyenne plus basse de 0.02 points pour KNN. L'accuracy est elle plus élevée de 0.04 pour le SVC. KNN semble cependant être le classifieur permettant d'obtenir l'écart entre les valeurs de rappel et de précision le plus bas pour chaque classe.

GLUTEN FREE

	Précison	Rappel	F-mesure	Support
0	0.74	0.66	0.70	47
1	0.81	0.86	0.83	78
Accuracy			0.78	125
Macro AVG	0.77	0.76	0.76	125
Micro AVG	0.78	0.78	0.78	125

c) DTC

Voici la matrice de confusion et le tableau présentant les mesures de tests pour le modèle DTC:



Avec ce modèle, la classe majoritaire des recettes sans-gluten est la mieux reconnue des deux. L'écart entre précision et rappel est moins important que pour les autres modèles y compris celui du KNN qui était déjà assez bas par rapport au SVC.

	Précison	Rappel	F-mesure	Support
0	0.64	0.62	0.63	47
1	0.78	0.79	0.78	78
Accuracy			0.73	125
Macro AVG	0.71	0.71	0.71	125
Micro AVG	0.73	0.73	0.73	125

d) Conclusion

Voici un tableau récapitulatif des macro f-mesure des trois modèles:

	SVC	KNN	DTC
Macro-AVG	0.78	0.76	0.71

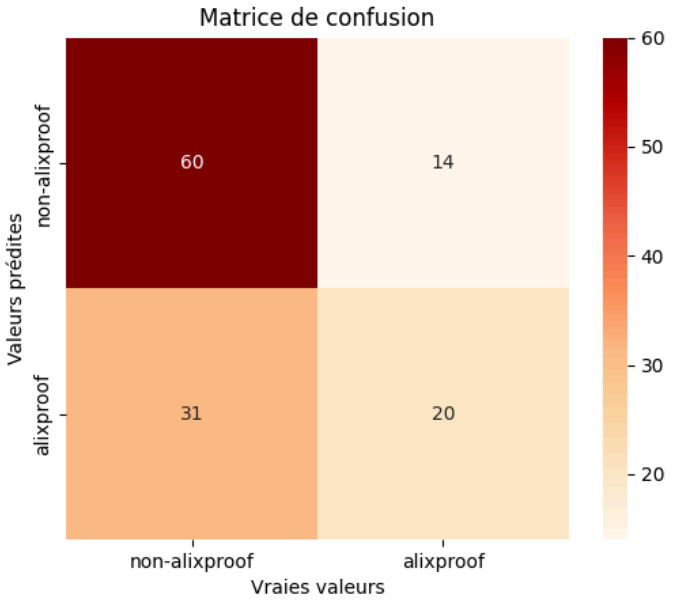
Le modèle qui reconnaît le mieux les recettes sans gluten est le modèle des SVM. Il y à un écart plus important entre précision et rappel mais pour toutes les autres mesures y compris l'accuracy et la précision, SVM est le modèle le plus performant pour les deux classes.

5.5. Alix Proof

Le régime "ne pas faire gonfler Alix" consiste à ne pas manger de produits provoquant des réactions inopinées sur le système inflammatoire d'Alix, autrement dit ne déclenche pas ses mastocytes.
Ce régime exclut: les produits laitiers, les œufs, les pommes de terres, les fruits de mer, le poisson etc.
L'équilibre entre les classes du régime Alixproof est le plus équilibré de tous nos régimes. Nous allons donc comparer les résultats de nos modèles avec leur micro-moyenne.

a) SVC

Voici la matrice de confusion et le tableau présentant les mesures de tests pour les SVM:

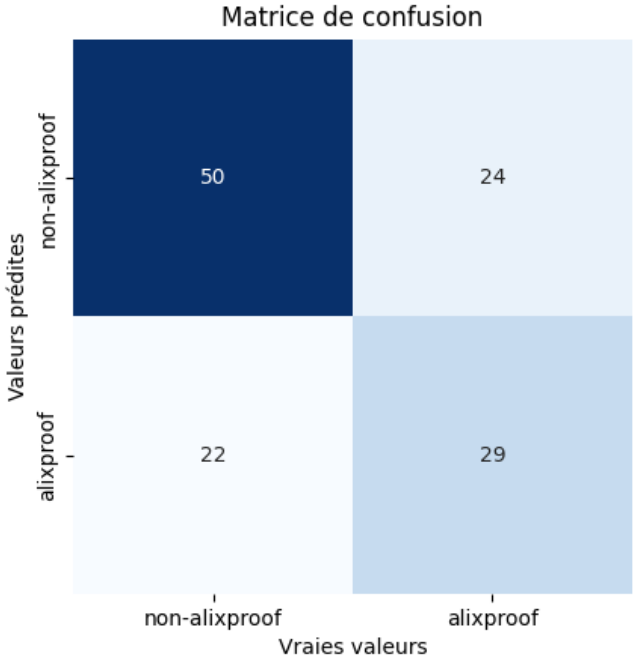


Les résultats du classifieur SVC sont parmi les moins bons obtenus jusqu'ici. Nous constatons notamment une f-mesure de 0.47 pour les recettes alixproof montrant que moins de la moitié des recettes de cette classe sont bien reconnues. Les valeurs de précision et de rappel sont elles également basses pour chaque classe (outre la valeur de rappel de 0.81 pour la classe non alixproof). Les macro et micro moyennes sont elles similaires grâce à l'équilibre des classes.

	Précison	Rappel	F-mesure	Support
0	0.66	0.81	0.73	74
1	0.59	0.39	0.47	51
Accuracy			0.64	125
Macro AVG	0.62	0.60	0.60	125
Micro AVG	0.63	0.64	0.62	125

b) KNN

Voici la matrice de confusion et le tableau présentant les mesures de tests pour le modèle KNN:

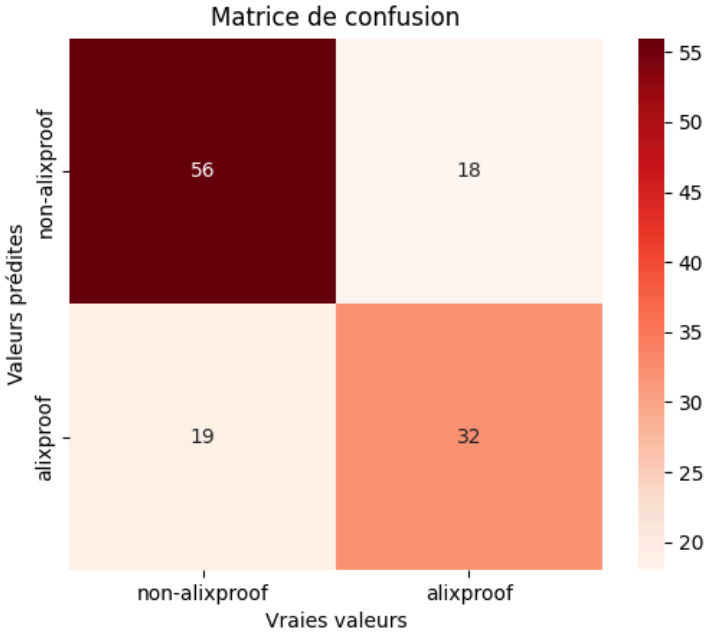


Les résultats de KNN sont équivalents à ceux obtenus pour le SVC mais présentent comme très souvent un meilleur équilibre entre les valeurs de précision et de rappel.
Un équilibre s'observe également entre les f-mesures de chaque classe ayant une différence d'uniquement 0.12 points contre 0.26 pour le SVC.

	Précison	Rappel	F-mesure	Support
0	0.69	0.68	0.68	74
1	0.55	0.57	0.56	51
Accuracy			0.63	125
Macro AVG	0.62	0.62	0.62	125
Micro AVG	0.63	0.63	0.63	125

c) DTC

Voici la matrice de confusion et le tableau présentant les mesures de tests pour le modèle DTC:



Ce modèle est celui qui reconnaît le mieux les recettes alixproof: avec 32 vrais positifs contre 20 et 29 recettes reconnues pour les SVM et KNN.

En plus de valeurs supérieures obtenues contrairement aux classifieurs KNN et SVC, le DTC semble offrir les résultats avec des écarts les moins élevés inter et intra classes. Nous avons effectivement un écart de 0.12 points entre les deux classes (similaire à KNN) mais des écarts d'uniquement 0.01 points entre les valeurs de précision et de rappel de chaque classe contre 0.02 pour le rappel et la précision de la classe alixproof pour KNN. Si ces valeurs tendent à varier dépendamment du split effectué, il semblerait que cette tendance reste consistante pour la plupart des essais.

	Précison	Rappel	F-mesure	Support
0	0.75	0.76	0.75	74
1	0.64	0.63	0.63	51
Accuracy			0.70	125
Macro AVG	0.69	0.69	0.69	125
Micro AVG	0.70	0.70	0.70	125

d) Conclusion

Voici un tableau récapitulatif des macro f-mesure des trois modèles:

	SVC	KNN	DTC
Micro-AVG	0.62	0.63	0.70

Malgré une distribution des classes équilibrés, les différences entre les recettes alixrpoof et non alixproof semblent ne pas être reconnues. Le DecisionTreeClassifier est le modèle qui les reconnait le mieux avec une micro-f-mesure de 0.70.

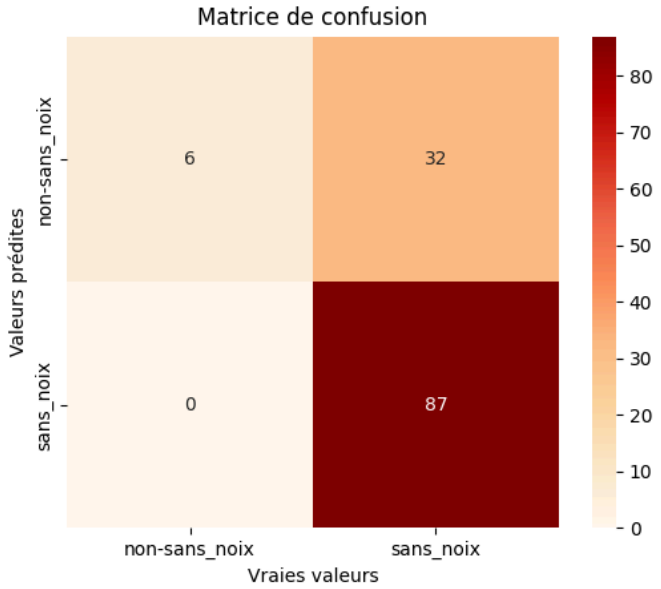
5.6. Sans Noix

Le régime sans noix s'adresse principalement aux allergiques aux noix. Les noix sont l'une des allergies les plus courantes avec le lait. Cette allergie peut provoquer des réactions importantes comme des chocs anaphylactiques provoquant des risques mortels pour la personne allergique.

Dans le cas du régime sans noix, nous souhaitons privilégier un modèle ne reconnaissant pas un nombre élevé de faux positifs (l'allergie aux noix pouvant s'avérer mortelle). Les deux métriques qui semblent donc ici les plus pertinentes sont donc la précision de la classe 1 et le rappel de la classe 0 , La précision de la classe 1 cherche à évaluer la capacité du modèle à reconnaître des résultats pertinents et corrects et le rappel de la classe 0 cherche à évaluer la capacité du modèle à ne pas passer à côté de recette contenant des noix.

a) SVC

Voici la matrice de confusion et le tableau présentant les mesures de tests pour les SVM:

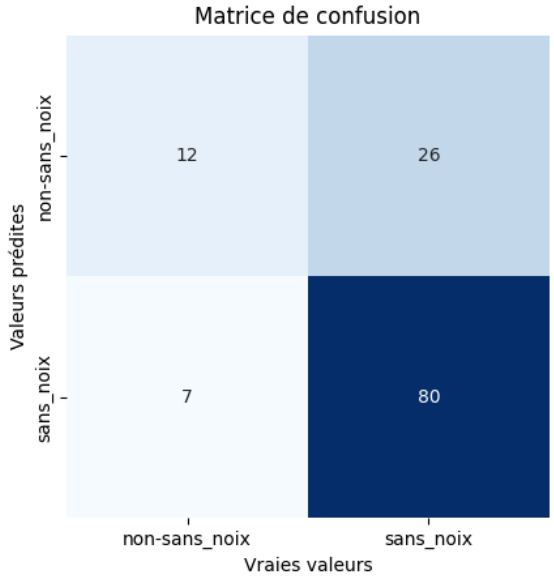


Les résultats obtenus ici montrent que l'ensemble des recettes sans noix ont correctement été reconnues. Cependant, nous faisons face à un nombre élevé de faux positifs, ce que nous cherchons à tout prix à éviter dans le cadre de ce régime. Nous préférons avoir une précision de classe 1 plus élevée au dépend de sa valeur de rappel.

	Précison	Rappel	F-mesure	Support
0	1.00	0.16	0.27	38
1	0.73	1.00	0.84	87
Accuracy			0.74	125
Macro AVG	0.87	0.58	0.56	125
Micro AVG	0.81	0.74	0.67	125

b) KNN

Voici la matrice de confusion et le tableau présentant les mesures de tests pour le modèle KNN:

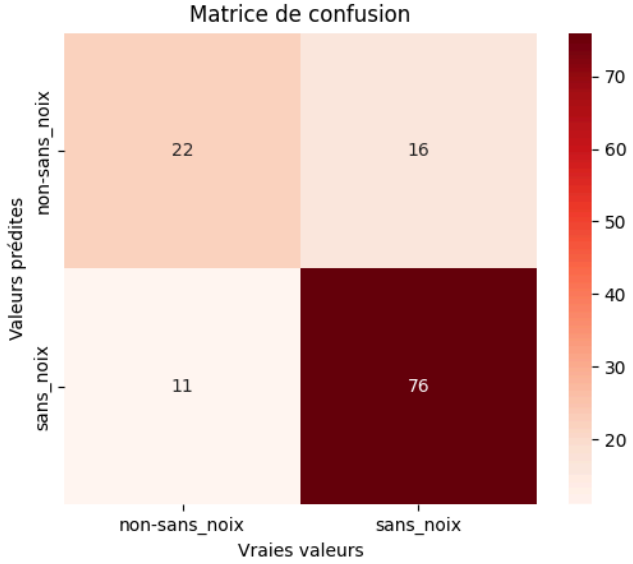


Les résultats obtenus avec KNN sont plus prometteurs que ceux du SVC. Nous avons en effet ici un nombre faux positifs moins élevé même si le rappel de la classe 0 et la précision de la classe 1 restent inquiétants.

	Précison	Rappel	F-mesure	Support
0	0.63	0.32	0.42	38
1	0.75	0.92	0.83	87
Accuracy			0.74	125
Macro AVG	0.69	0.62	0.63	125
Micro AVG	0.72	0.74	0.70	125

c) DTC

Voici la matrice de confusion et le tableau présentant les mesures de tests pour le modèle DTC:



Enfin, les résultats obtenus avec le classifieur DTC montrent qu'il s'agit du meilleur modèle jusqu'ici. Les deux valeurs d'intérêt, rappel de la classe 0 et précision de la classe 1, ont augmenté de 0.25 et 0.08 points respectivement. Les valeurs moyennes sont elles également meilleures que pour SVC et KNN. Le nombre de faux positifs, si plus bas, reste malgré tout trop élevé.

	Précison	Rappel	F-mesure	Support
0	0.67	0.58	0.62	38
1	0.83	0.87	0.85	87
Accuracy			0.78	125
Macro AVG	0.75	0.73	0.73	125
Micro AVG	0.78	0.78	0.78	125

d) Conclusion

Voici un tableau récapitulatif des macro f-mesure des trois modèles:

	SVC	KNN	DTC
Précision 1	0.73	0.75	0.83
Rappel 0	0.16	0.32	0.58

La classe avec noix est la classe minoritaire, il est donc normal qu'elle soit moins reconnue.
Pour ce test, nous cherchons un modèle qui soit le plus précis possible quand il est question de sans-noix et détecte le mieux les recettes avec noix, une précision alors élevée pour la classe 1.
Le meilleur modèle pour cette tâche est le DTC mais il n'est pas suffisamment performant pour ne pas risquer un choc anaphylactique.

Conclusion

Voici donc un tableau récapitulatif des modèles plus performants pour chaque régime alimentaire avec leur macro f-mesure:

	Végétarien	Végan	Crudivore	Sans-gluten	Alixproof	Sans noix
SVM				**0.78**		
KNN			**0.83**			
DTC	**0.75**	**0.73**	**0.83**		**0.70**	**0.73**

On voit que généralement le DecisionTreeClassifier est celui qui reconnaît le mieux les régimes. Cependant il est important de noter que pour la plupart de nos régimes les modèles KNN et DTC avaient des valeurs similaires, les départager a été possible en admettant que nous priorisions certains éléments à d'autres dont notamment un équilibre précision/rappel ou une valeur plus élevée de macro-moyenne.

Le régime qui à les mesures les plus élevées est le régime **crudivore (0.83) avec le modèle DTC**. Les régimes les moins bien reconnus sont **alixproof (0.70) et vegan (0.73).** Ces deux régimes ont comme dénominateur commun les produits laitiers et les œufs.

À l'aide d'une des fonctions des scripts d'entraînement nommée ***obtention_phrases***, nous avons été dans la capacité d'obtenir les recettes placés dans chaque grande catégorie (vrais positifs, faux positifs, faux négatifs, vrais négatifs) résultant de l'entraînement pour chaque modèle et régime alimentaire. Ces phrases et leurs catégories correspondantes sont stockées dans les fichiers ***csv*** du dossier ***data.*** Ces fichiers nous permettent d'obtenir de précieuses informations sur les termes et n-grammes pouvant être responsables d'une bonne ou mauvaise classification de chaque modèle en fonction du régime.

- **Crudivore :**

Le régime crudivore est donc celui étant le mieux reconnu par nos modèles malgré sa représentation réduite dans notre corpus. Les mots et n-grammes les plus présents dans les catégories des vrais positifs et vrais négatifs sont eux attendus pour ce régime spécifique. Les recettes des ***vrais négatifs*** contiennent des mots tels que ***cuisson***, ***cuire***, ***four***, ***poêle*** et ***fumer*** complètement ***absents des recettes des vrais positifs.***

- **Faux positifs :**

  Il semblerait que les recettes reconnues comme crudivore mais ne l'étant pas réellement sont des recettes requièrant un moyen de cuisson moins courant. Nous pouvons notamment citer l'exemple suivant :

  1 min Étape 1 Laver et couper le concombre en 4 dans le sens de la longueur puis en petits triangles. Étape 2 Mettre le vinaigre et le sucre 30 sec au ***micro-ondes*** *(jusqu'à obtenir un mélange sirupeux et aigre-doux).*

L'utilisation du micro-onde et non pas du four ou d'une poêle, ustensiles bien plus courants dans notre corpus, semble empêcher notre modèle de le reconnaître comme un moyen de cuisson et fait donc de cette recette une recette prédite comme crudivore. L'unité utilisée pour la cuisson (ici des secondes) est également généralement absente des recettes au profit de minutes ou d'heures.

- **Faux négatifs :**
    1. **Verbes souvent liés à la cuisson :** il semblerait que parmi les recettes non reconnues comme crudivores, de nombreux continent des verbes généralement utilisés lors de recettes avec cuisson tels que "arrosez" pour arroser des ingrédients avant cuisson, "étaler" pour étaler une pâte sur une plaque de cuisson ou "mettre" pour mettre quelque chose dans un plat allant au four.

       La recette : Couper au rasoir à légumes les courgettes non épluchées, les mettre en alternant les couleurs dans le plat de service, ***arroser*** *de jus de citron, d'huile d'olive, saler, poivrer, ajouter de la coriandre en grains concassés et de la coriandre fraîche, des zestes de citron finement hachés et une gousse d'ail pressée.*

    2. **Noms liés à la cuisson :** il semblerait également que beaucoup de ces recettes contiennent des noms utilisés pour parler de cuisson comme "température" ou bien "moule". Les recettes nécessitant un repos à température ambiante ou bien un moule allant au frigo sont donc parfois mal reconnues.

       Ingrédients : Pour 4 personnes 1/2 pastèque moyenne QS de feta, environ 4 bâtonnets Des olives noires à la grecque Quelques feuilles de roquette Pour la sauce: 10 cl de lait de coco à ***température ambiante*** *1 grosse poignée de feuilles de coriandre fraîche sel et poivre*

    3. **Référence à d'autres aliments souvent cuits :** certains aliments présents dans des recettes de la catégorie faux négatifs sont généralement cuits ou utilisés dans des recettes nécessitant une cuisson. Nous avons par exemple la mention de "chair" d'amandes souvent utilisé pour de la viande.

       Ouvrez les amandes avec un couteau à huîtres et prélevez la chair. *Gardez les parties fermes et rincez bien sous l'eau froide.*

- **Alixproof:**

Le régime Alixproof est le moins bien reconnu par le modèle bien qu'il soit celui dont la répartition des classes dans le corpus est la plus équilibrée.

- **Faux positifs:**
  Dans les faux positifs on retrouve beaucoup d'allergènes tels quels dans les listes d'ingrédients: oeufs, thon, feta, lait, pomme de terre, huîtres, crème, yaourt, etc. Pour des allergènes qui ne sont pas très présents dans le corpus comme les huîtres ou des types particuliers de fromage comme la feta on peut l'expliquer par la rareté des ces occurrences mais pour le reste des faux positifs, il faudrait faire plus de tests.

- **Faux négatifs:**
  Le modèle peut classer les recettes comme non Alixproof alors qu'elles le sont pour plusieurs raisons:
  1. Les mentions d'alternatives aux produits allergènes comme : "mayonnaise vegan", "crème de soja", "lait de coco", "yaourt de soja" etc.
  2. Les mentions à des produits qui portent le même nom que les allergènes : "œufs de pâques", "crème balsamique", "fécule de maïs" ou "fécule de pomme de terre", des recettes salées avec des "pommes", etc. Il se pourrait qu'il confond les moules à gâteau avec les fruits de mer du même nom.
  3. Les glaces semblent être cataloguées non Alixproof, que ce soit le "sucre glace" ou même des "glaces à l'eau". Dans les vrais positifs, on ne retrouve aucune recette de glace ou recette mentionnant le mot "glace".
  4. On retrouve aussi des recettes salées comme des boeufs bourguignon ou des currys, qui contiennent souvent des pommes de terres ou sont servis avec.

Grâce aux fichiers analysés, nous avons pu commencer à comprendre la manière dont les modèles arrivent à faire des choix. Nous pensons que pour des cas similaires au régime Alixproof, nos modèles pourraient bénéficier d'un apprentissage avec contraintes et donc de règles prédéfinies comme nos listes d'allergènes par exemple. Ces résultats sont intéressants car opposés à nos hypothèses initiales. Le régime crudivore est le mieux reconnu et le vegan l'un des moins bien reconnu, complètement l'inverse de ce que nous avions prévu. Il semblerait que dans notre cas, la diversité des attributs discriminants ait plus de poids que l'équilibre des classes.

Un défi lorsque l'on travaille avec des situations médicales pouvant être dangereuses comme les allergies aux noix, est de privilégier la sûreté au reste. En effet, il serait bon lors de l'apprentissage de faire que si le modèle à le moindre doute ou hésite entre deux classes, la classe par défaut serait "potentielle présence de noix on ne prend pas de risque".

Ce qui est en revanche intéressant pour ce type de modèle , est que l'on ne les mesure pas selon une mesure globale portant sur les deux classes mais sur le rappel de la classe "présence de l'allergène" et sur la précision de la classe "recette compatible avec le régime". Dans ce cas précis, il vaut mieux avoir des valeurs "Faux Négatifs" que des valeurs "Faux positifs" pour ne pas risquer d'empoisonner quelqu'un.

Concernant les perspectives futures pouvant inclure des modèles de prédiction similaires, nous avons pensé à plusieurs applications possibles de notre corpus d'apprentissage. Dans un contexte hospitalier, un modèle plus performant pourrait être chargé de la création automatique de menus admettant des contraintes alimentaires. Les sites de recettes pourraient également être capables de catégoriser plus facilement leurs recettes.

Options supplémentaires de l'apprentissage:

- Création de menus journaliers pour répondre à la question du sain et de l'équilibre alimentaire avec les besoins journaliers d'une personne.
- Voir un régime sur une journée ou une semaine permettrait aussi de prendre en compte des pathologies ayant des besoins spécifiques: lisser la glycémie pour les personnes atteintes de diabète. Sur une semaine, le potentiel inflammatoire des aliments pourrait aussi être analysé pour des pathologies nécessitant un régime anti-inflammatoire.