Nom : LE Prénom : Ba Minh

Document à déposer au format PDF uniquement sur l'espace de cours moodle

avant le lundi 25 mars à 8h00. Le document doit être nommé nom-prenom.pdf
Vérifiez au préalable que votre copie numérique au format PDF coïncide bien avec la présente copie. Ne pas oublier de remplir l'entête de document ci-dessus avec vos nom et prénom.

Composition des aliments

Nous considérons un ensemble de données constitué par l'agence américaine des aliments. Dans cet ensemble environ 8000 produits sont analysés et leur composition est enregistrée.

Importation et préparation des données

Charger et préparer les données en respectant scrupuleusement le code du notebook disponible sous ce lien https://colab.research.google.com/drive/1zc4ollC833_BCF2HlzaIg5e5kJGSrrj2?usp=sharing.

Vous pouvez enregistrer une copie dans votre drive pour travailler. Dans le dernier bloc du notebook, vous devez remplacer le numero d'étudiant par votre propre numéro d'étudiant.

A l'issue de l'exécution de de début de notebook, vous obtenez une table pandas nommée food_df.

1. Combien d'observations et combien de variables contient cette table ?

```
Répondre ici.
Nombre d'observations : 2000
Nombre de variables : 15
```

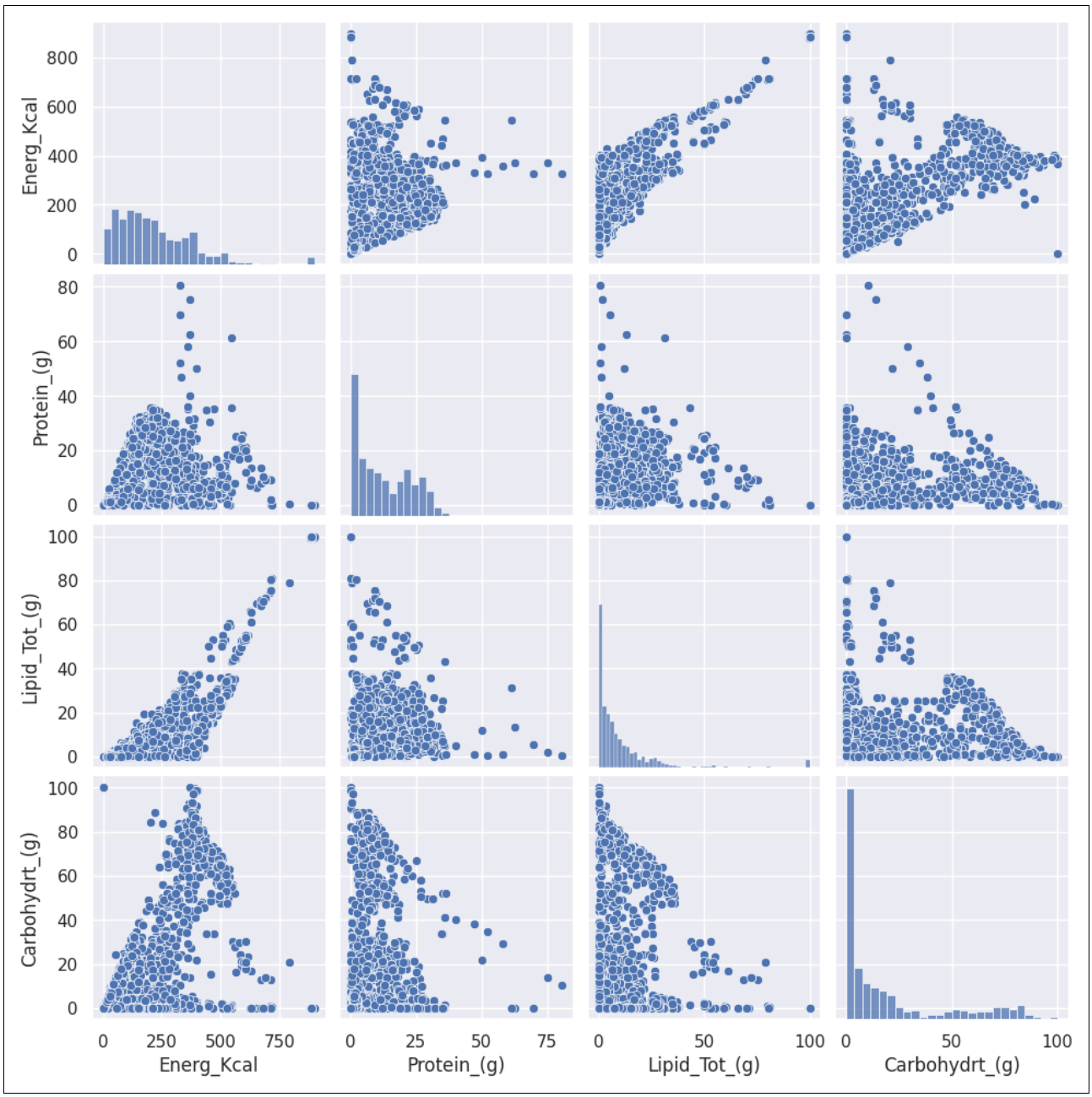
2. Donner le nom ainsi que les différentes valeurs possibles des 3 variables catégorielles.

Nom de la variable	Valeurs possibles
has_fiber	[0,1]
has_vitamine	[0,1]
sugar	['a lot','little','no']

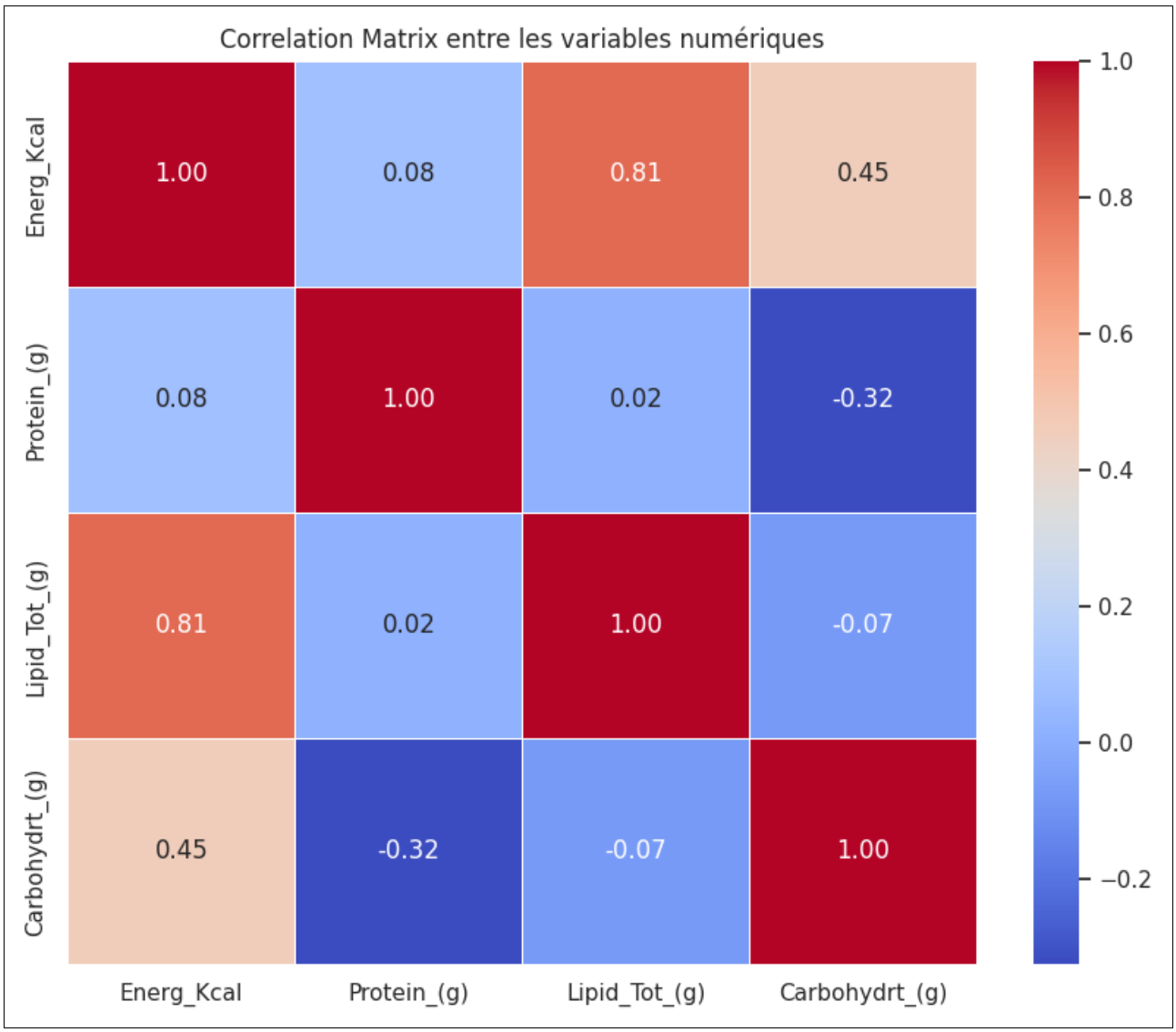
3. Effectuez une visualisation des données deux à deux. Donnez le nom de 2 variables qui semblent être intéressantes pour prédire le nombre de calories d'un aliment (colonne "Energ_Kcal") en justifiant.

```
Mettre le graphe ici.
```

Nom : LE Prénom : Ba Minh



Nom : LE Prénom : Ba Minh



Justifiez ici.
« Lipid_Tot_(g) » a une forte corr é lation positive avec Ener_Kcal(0.81). Cela signfie que plus un aliment contient de liquides, plus il est calorique. C'est logique car les lipids fournissement 9 kcal/g, ce qui en fait la macronutriment le plus é nerg é tique.
« Carbonhydrt_(g) » a une co rr é lation mod é ré e positive avec Energ_Kcal(0.45). Les glucides fournissement 4 kcal/g, et bien qu'ils ne soient pas aussi é nerg é tique que les lipids , il influencent é galement le nombre total de clories.
Lipid_Tot_(g) est la variable la plus fortement corrélée avec les calories, ce qui en fait un excellent prédicteur.
Carbohydrt_(g) présente aussi une corrélation significative, ce qui le rend pertinent pour affiner la prédiction.
Les deux variables les plus intéressantes pour prédire Energ_Kcal sont :
- Lipid_Tot_(g)
- Carbohydrt_(g)

Nom : LE Prénom : Ba Minh

4. Notre objectif est de prédire les calories d'un aliment (colonne "Energ_Kcal"). Centrez et réduisez les 11 variables numériques qui seront utilisées en entrée du modèle (toutes les colonnes numériques sauf "Energ_Kcal").

```
Copier votre code ici.
from sklearn.preprocessing import StandardScaler
columns_to_drop = ['has_fiber', 'has_vitamine', 'sugar']
for col in columns_to_drop:
    if col in food_df.columns:
        food_df = food_df.drop(columns=[col])
    else:
        print(f"Column '{col}' ")
num_vars = food_df.select_dtypes(include=["number"]).columns
num_vars = num_vars.drop("Energ_Kcal", errors='ignore')
scaler = StandardScaler()
food_df[num_vars] = scaler.fit_transform(food_df[num_vars])
print("Colonnes après transformation : ", food_df.columns)
print(food_df[num_vars].describe())
```

5. Nous souhaitons maintenant transformer les variables catégorielles. Utilisez la fonction *pd.get_dummies* afin d'encoder les variables catégorielles. Choisissez bien la valeur de l'argument drop_first afin d'obtenir 16 variables dans le dataframe final.

```
Copier votre code ici.
categorical_columns = ['has_fiber', 'has_vitamine', 'sugar']
food_df_encoded = pd.get_dummies(food_df,
columns=categorical_columns, drop_first=True)
print("Shape après l'encodage des variables catégorielles :",
food_df_encoded.shape)
```

6. Afin de terminer avec la préparation de données, nous souhaitons obtenir un vecteur y (target variable) avec les calories de chaque aliment et une matrice X (features) contenant toutes les caractéristiques de chaque aliment. Enfin, séparez le jeu de données et gardez 15 % du jeu de données pour le test (que l'on nommera X_test et y_test) et les 85 % restant pour l'entraînement et la validation (que l'on nommera X_no_test et y_no_test). Donnez le nombre de lignes de l'ensemble de test.

```
Copier ici votre code.
from sklearn.model_selection import train_test_split
x = food_df_encoded.drop(columns=["Energ_Kcal"])
y = food_df_encoded["Energ_Kcal"]
x_no_test, x_test, y_no_test, y_test = train_test_split(x, y,
test_size=0.15, random_state=42)
print("Nombre de lignes dans l'ensemble de test :",
x_test.shape[0])
```

```
Répondez ici.
Nombre de lignes dans l'ensemble de test : 300
```

Nom : LE Prénom : Ba Minh

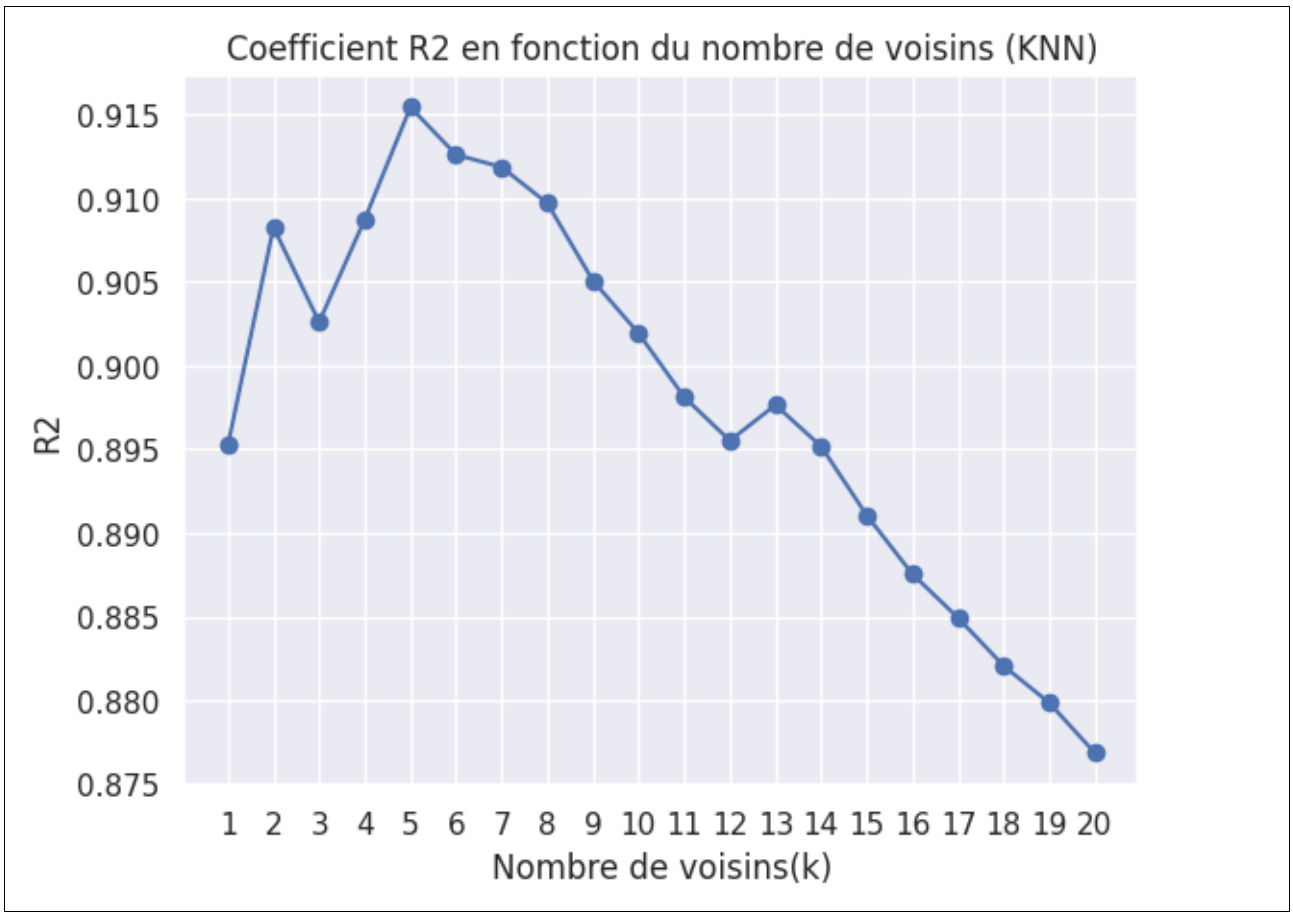
Utilisation de l’algorithme des k plus proches voisins

7. Nous allons prédire le nombre de calories des aliments en utilisant la méthode des k plus proches voisins. Nous souhaitons déterminer le nombre de voisins donnant le meilleur modèle par rapport à la métrique « R2 ». Pour cela, nous avons choisi de faire de la validation train-validation-test split en utilisant 20 % du jeu de données d’entraînement (X_no_test et y_no_test) pour la validation. Faîtes un graphique donnant la valeur du coefficient « R2 » en fonction du nombre de voisins.

```
Copier ici votre code.
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(x_no_test,
y_no_test, test_size=0.2, random_state=42)
k_values = range(1, 21)
mean_squared_errors = []
for k in k_values:
    knn = KNeighborsRegressor(n_neighbors=k)
    knn.fit(x_train, y_train)
    y_pred = knn.predict(x_val)
    r2 = mean_squared_error(y_val, y_pred)
    mean_squared_errors.append(r2)
plt.plot(k_values, mean_squared_errors, marker='o')
plt.xlabel('Nombre de voisins(k)')
plt.ylabel('R2')
plt.title('Coefficient R2 en fonction du nombre de voisins (KNN)')
plt.grid(True)
plt.show()
```

```
Copier ici le graphique.
```

Nom : LE Prénom : Ba Minh



8. Quel est le nombre de voisins optimaux et quel est le score « R2 » de la méthode des k plus proches voisins avec le nombre de voisins optimaux en entraînant le modèle avec X_no_test et y_no_test et en calculant le score sur X_test et y_test ?

```
Copier ici votre code.
optimal_k = k_values[np.argmax(r2_scores)]
knn_optimal = KNeighborsRegressor(n_neighbors=optimal_k)
knn_optimal.fit(x_no_test, y_no_test)
r2_test = knn_optimal.score(x_test, y_test)
print(f"Le nombre optimal de voisins est : {optimal_k}")
print(f"Le score R2 sur l'ensemble de test est : {r2_test}")
```

```
Mettez votre réponse ici.
Le nombre optimal de voisins est : 5
Le score R2 sur l'ensemble de test est : 0.8858553825262918
```

Utilisation d’une forêt aléatoire

9. Nous proposons maintenant de prédire le nombre de calories des aliments en utilisant une forêt aléatoire. Nous souhaitons déterminer les hyper-paramètres donnant le meilleur modèle par rapport à la métrique « R2 ». Pour cela, nous avons choisi de faire de faire de la validation croisée 5-Folds sur le jeu de données d’entraînement (X_no_test et y_no_test). Utilisez la fonction *GridSearchCV*

de scikit-learn afin de faire varier au minimum 4 hyper-paramètres de la méthode. Quels sont les meilleures valeurs des hyper-paramètres?

```
Copier ici votre code.
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
param_grid = {
    'n_estimators': [50,100,200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
rf = RandomForestRegressor(random_state=42)
grid_search  =  GridSearchCV(estimator=rf,  param_grid=param_grid,
cv=5, scoring='r2',n_jobs=-1)
grid_search.fit(x_no_test, y_no_test)
best_params = grid_search.best_params_
print("Les meilleurs valeurs des hyper_paramètres :")
print(best_params)
```

```
Mettez votre réponse ici.
Les meilleurs valeurs des hyper_paramètres :
{'max_depth': 20, 'min_samples_leaf': 2, 'min_samples_split': 2,
'n_estimators': 200}
```

10. Quel est le score « R2 » de la forêt aléatoire avec les meilleurs hyper-paramètres de la question précédente en entraînant le modèle avec X_no_test et y_no_test et en calculant le score « R2 » sur X_test et y_test ? Quelle méthode entre les k plus proches voisins et la forêt aléatoire obtient les meilleures performances?

```
Copier ici votre code.
rf_optimal          =          RandomForestRegressor(max_depth=20,
min_samples_leaf=2,          min_samples_split=2,n_estimators=200,
random_state=42)
rf_optimal.fit(x_no_test, y_no_test)
r2_test = rf_optimal.score(x_test, y_test)
print(f"Le score R2 de la forest aléattoire avec les meilleurs
hyper-paramètres test est : {r2_test:.4f}")
```

```
Mettez votre réponse ici.
Le score R2 de la forest aléattoire avec les meilleurs hyper-
paramètres test est : 0.9952
```

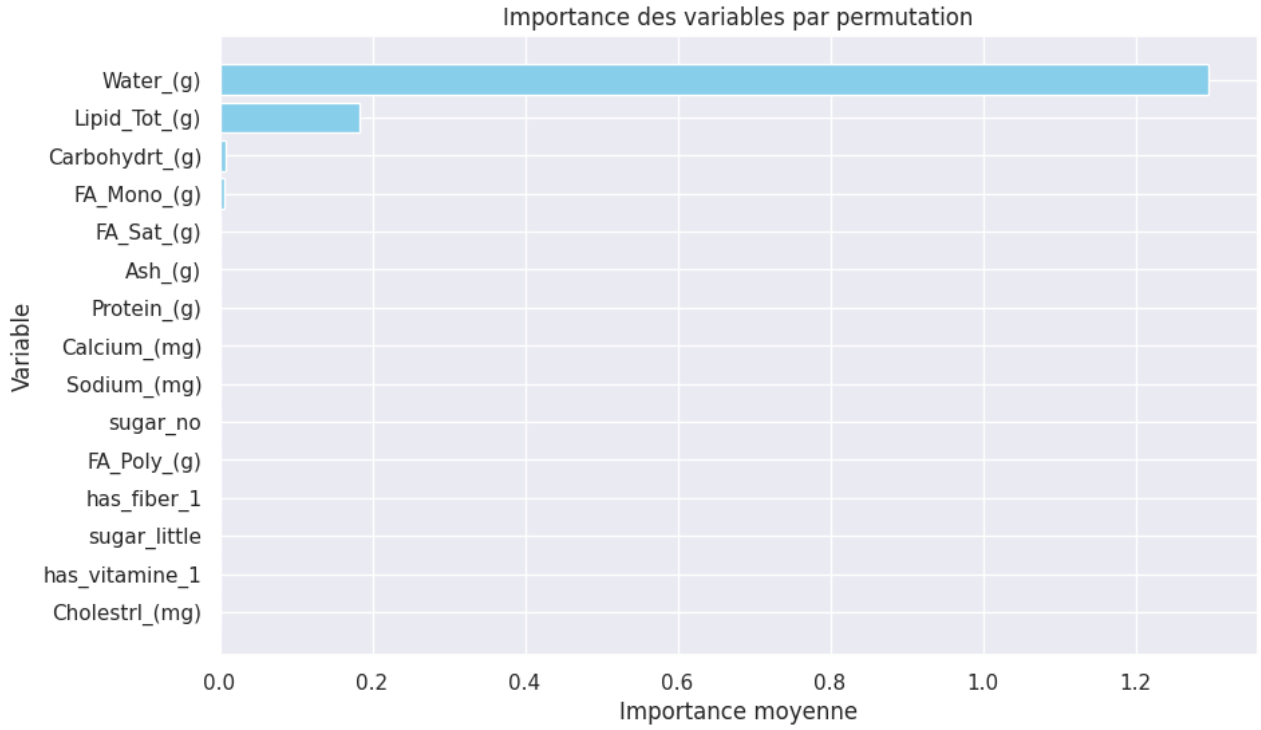
11. Pour terminer, afin d'obtenir une meilleure interprétation de la forêt aléatoire, nous souhaiterions savoir quelles sont les variables qui influent le plus sur la prédiction. En utilisant la fonction *permutation_importance* de scikit-learn, réalisez un graphique donnant l'importance de chaque variable dans la prédiction. Quelles sont les variables qui ont le plus d'importance?

```
Copier ici votre code.
from sklearn.inspection import permutation_importance
```

Nom : LE Prénom : Ba Minh

```
result = permutation_importance(rf_optimal, x_test, y_test,
n_repeats=10, random_state=42, n_jobs=-1)
importance_scores = result.importances_mean
importance_df = pd.DataFrame({'Variable': x_test.columns,
'Importance': importance_scores})
importance_df = importance_df.sort_values(by='Importance',
ascending=False)
plt.figure(figsize=(10, 6))
plt.barh(importance_df['Variable'],
importance_df['Importance'],color = 'skyblue')
plt.xlabel('Importance moyenne')
plt.ylabel('Variable')
plt.title('Importance des variables par permutation')
plt.gca().invert_yaxis()
plt.show()
print("Les 5 variables les plus importantes sont :")
print(importance_df.head())
```

Copier ici le graphique.



Mettez votre réponse ici.

Les 5 variables les plus importantes sont :

	Variable	Importance
0	Water_(g)	1.294894
2	Lipid_Tot_(g)	0.183946
4	Carbohydrt_(g)	0.007313
8	FA_Mono_(g)	0.006779
7	FA_Sat_(g)	0.001266

Nom : LE Prénom : Ba Minh

12. Que pouvez-vous conclure sur le TP (capacité de prédiction des calories, importance des variables, choix des modèles, …) ?

Mettez votre réponse ici.
+, Capacité de prédiction des calories :
- Forêt aléatoire : Avec un score R^2 de 0.9952, ce modèle montre une excellente précision pour prédire le nombre de calories.
- K plus proches voisins (KNN) : Le meilleur modèle KNN atteint un score R^2 de 0.8859, ce qui est acceptable mais moins performant que la forêt aléatoire.
- ➔ La forêt aléatoire se révèle donc être le modèle le plus performant pour cette tâche.

+, Importance des variables :
- Water_(g) est la variable la plus influente, ce qui peut être contre-intuitif mais s'explique par le fait que les aliments riches en eau sont généralement moins caloriques.
- Lipid_Tot_(g) est un facteur clé.
- Carbohydrt_(g) joue également un rôle.
- FA_Mono_(g) et FA_Sat_(g) ont une importance plus faible mais non négligeable.

+, Choix des modèles :
- KNN est une approche simple, mais elle est limitée en haute dimension et sensible aux données bruitées.
- La forêt aléatoire s'est avérée être le meilleur modèle, grâce à sa capacité à gérer des relations complexes entre les variables et à donner une interprétation via l'importance des caractéristiques.
- L'optimisation des hyper-paramètres a permis d'améliorer la performance du modèle de forêt aléatoire.

+, Enseignements généraux :
- La normalisation des données a été essentielle pour assurer une bonne convergence des modèles.
- L'encodage des variables catégorielles a permis d'intégrer toutes les informations disponibles.
- La validation croisée a aidé à identifier les meilleurs hyper-paramètres et à éviter le sur-apprentissage.

Conclusion :
La forêt aléatoire est clairement le modèle à privilégier pour prédire les calories des aliments, avec un excellent score R^2. L'analyse a également mis en évidence l'importance des lipides et des glucides, mais aussi le rôle de la teneur en eau