Royaume du Maroc
UNIVERSITE MOHAMED V AGDAL - RABAT
Ecole Mohammadia d'Ingénieurs

Departement Modélisation et Informatique Scientifique

Rapport projet Machine Learning

Encadré par :

- Mr.y.lamrani

Fait par:

- ARFAOUI Abderrahim

Année Universitaire :2024/2025

Table des contents:

REMERCIEMENTS

Je tiens à exprimer mes sincères remerciements à mon professeur, Mr .Youssef Lamrani , professeur de l'enseignement supérieur à l'EMI en Génie Modélisation et Informatique Scientifique, pour m'avoir permis d'explorer un sujet aussi enrichissant que la prédiction . Cette approche de machine learning consiste à établir une relation entre une variable cible et une ou plusieurs variables explicatives, afin de prédire des valeurs futures ou de mieux comprendre les facteurs qui influencent un phénomène.

Grâce à ce projet, j'ai pu me familiariser avec les étapes essentielles du machine learning, telles que la préparation des données, la construction et l'évaluation de modèles, et la création d'une **API en Flask** permettant de déployer ces prédictions. Cela m'a offert l'opportunité d'approfondir mes compétences et de comprendre l'importance de la régression linéaire dans des applications concrètes, où elle est utilisée pour modéliser et anticiper des résultats basés sur des données historiques.

Je voudrais également exprimer ma reconnaissance pour l'encadrement précieux de , Mr.Youssef Lamrani, ainsi que pour ses conseils avisés, qui ont éclairé mon travail .

1-Présentation du projet

Ce projet vise à prédire les charges de chauffage (y1) et de refroidissement (y2) pour des bâtiments résidentiels en fonction de huit caractéristiques architecturales. Le jeu de données utilisé pour cette analyse a été créé par Angeliki Xifara, ingénieure civile et en structures, et traité par Athanasios Tsanas, chercheur au Centre d'Industrial et Applied Mathematics de l'Université d'Oxford.

1-1-Description du Jeu de Données

Le jeu de données contient 768 échantillons, chacun représentant une forme de bâtiment différente, obtenue par simulation dans le logiciel Ecotect. Les bâtiments varient selon des paramètres tels que la surface vitrée, sa répartition et l'orientation du bâtiment, permettant une diversité de configurations.

Ces variations visent à obtenir des prédictions précises de la performance énergétique à partir de différentes caractéristiques architecturales.

1-2-Attributs

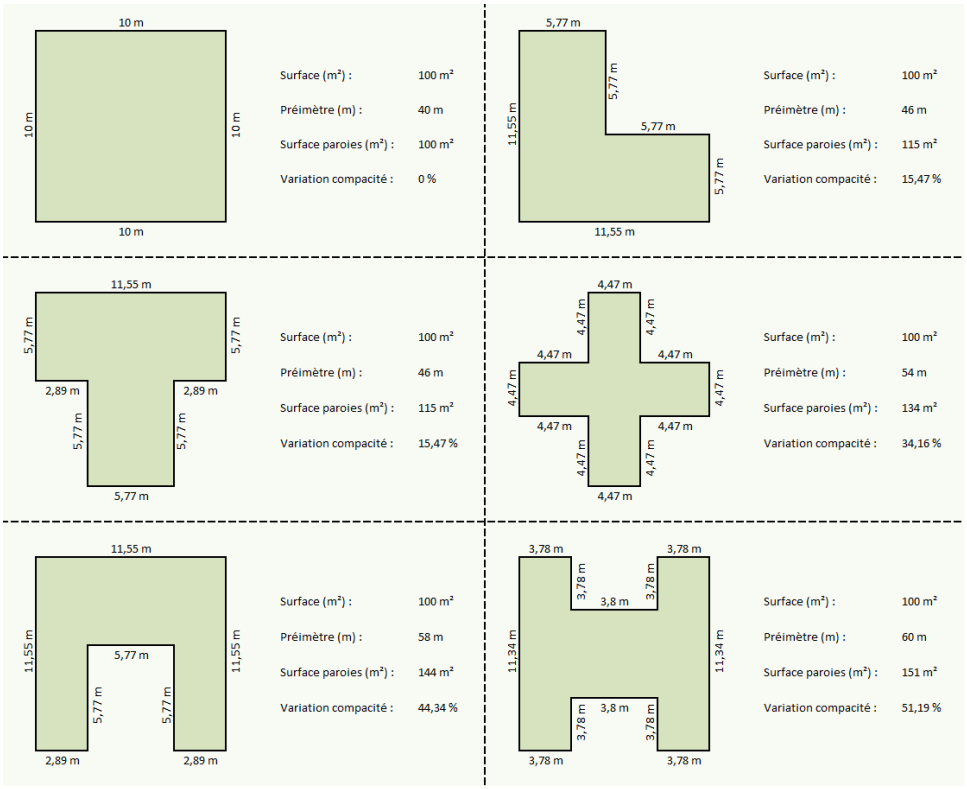
Le jeu de données contient huit attributs (ou **variables prédictives**), notés X1 à X8, et deux **variables cibles** notées y1 et y2. Ces variables sont décrites en détail ci-dessous :

X1 : Compacité relative (Relative Compactness)

La compacité thermique désigne la capacité d'un matériau ou d'une structure à minimiser les pertes de chaleur, favorisée par une forme et des dimensions optimisées pour réduire l'exposition aux fluctuations de température. En architecture, une maison bien compacte conserve mieux la chaleur, réduisant ainsi les besoins en chauffage et augmentant son efficacité énergétique. Pour maximiser la compacité thermique, il est crucial d'isoler adéquatement les murs, le toit et le plancher tout en utilisant des matériaux à haute performance thermique.

Exemple de calcule :

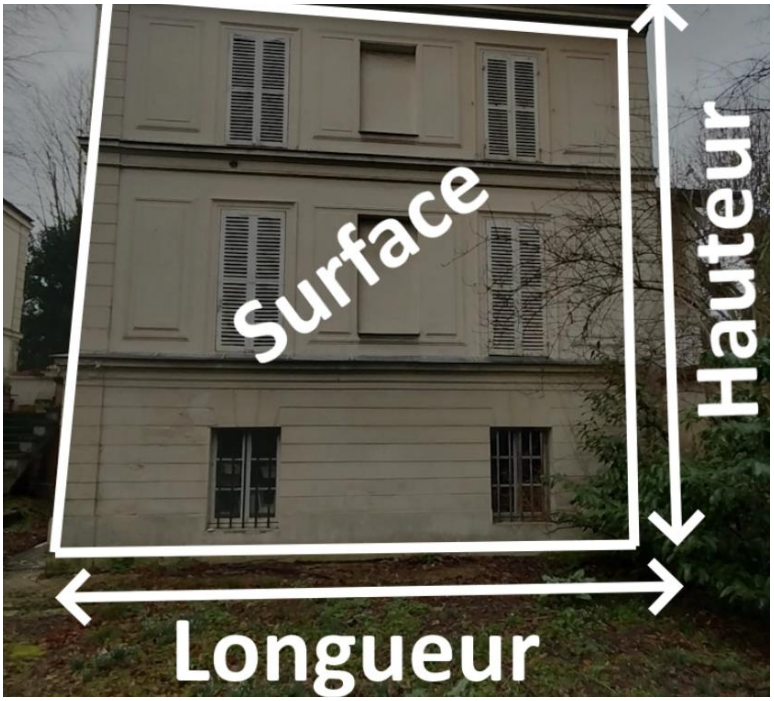
une même surface de plancher (100m²) peut avoir entre 100m² de surface de façade déperditive et 151m² de surface de façade déperditive. Ce qui représente une différence de plus de 50% de perte d'énergie.



Source :https://www.etude-bet.fr/Faq/Qu-est-ce-que-la-compacite-d-une-construction.

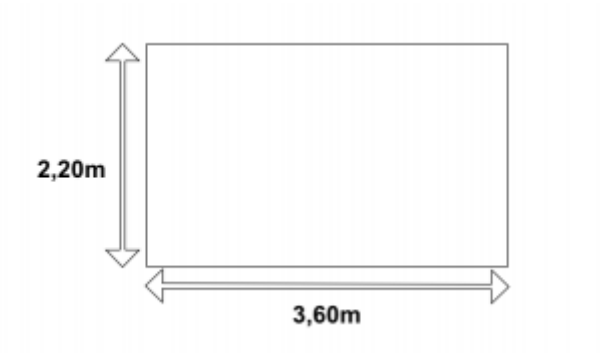
X2 : Surface de la façade (Surface Area)

La surface de la façade désigne la surface totale exposée à l'extérieur du bâtiment, influençant les échanges thermiques. Une surface plus grande entraîne en général une perte de chaleur plus élevée en hiver et un gain de chaleur



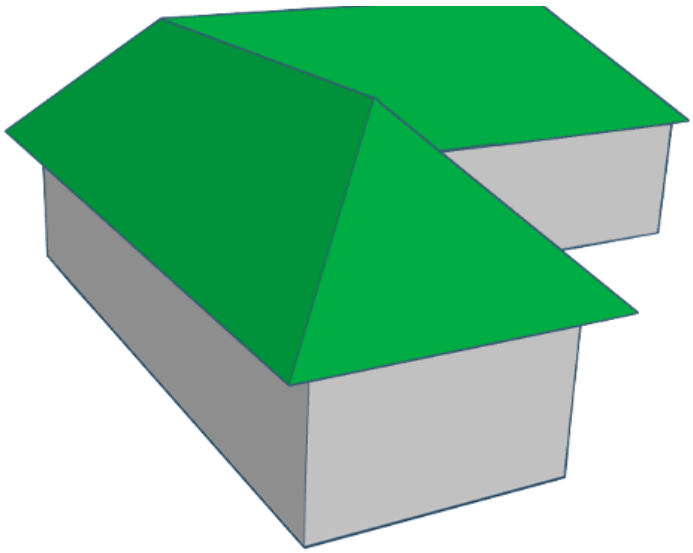
X 3 : Surface des murs (Wall Area)

Cette variable correspond à la surface totale des murs extérieurs du bâtiment. Elle influe directement sur les échanges thermiques, en fonction de l'isolation et de l'épaisseur des murs.



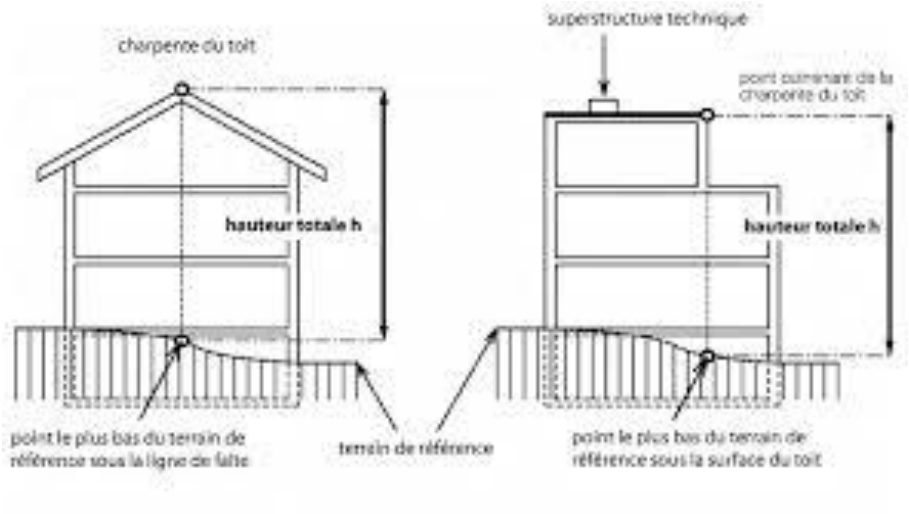
X 4 : Surface du toit (Roof Area)

La surface du toit détermine la quantité de chaleur qui peut être absorbée ou perdue par le toit. Un toit plus grand peut accroître les pertes de chaleur ou les gains thermiques selon la saison.



X 5 : Hauteur totale (Overall Height)

La hauteur totale du bâtiment influence la convection de l'air à l'intérieur. Par exemple, les bâtiments plus hauts peuvent bénéficier de la stratification de l'air, ce qui peut affecter les charges thermiques de chauffage et de refroidissement.



X 6 : Orientation (Orientation)

L'orientation du bâtiment, ou la direction dans laquelle il est exposé, joue un rôle majeur dans l'ensoleillement direct et les gains thermiques. Les bâtiments orientés vers le sud (dans l'hémisphère nord) reçoivent plus de lumière solaire,

X 7: Surface vitrée (Glazing Area)

La surface vitrée fait référence à la proportion de fenêtres ou d'autres surfaces transparentes du bâtiment. Une plus grande surface vitrée peut entraîner une augmentation des gains solaires mais aussi des pertes thermiques la nuit.

X 8 : Répartition de la surface vitrée (Glazing Area Distribution)

Cette variable indique la répartition des surfaces vitrées sur le bâtiment (par exemple, répartie uniformément ou concentrée sur une seule façade). La répartition peut influencer les gains thermiques solaires de manière significative.

1-3-Les variables cibles:

Y1 : Charge de chauffage (Heating Load)

La charge de chauffage représente l'énergie nécessaire pour maintenir une température intérieure confortable en hiver. Elle dépend directement de l'isolation du bâtiment et de sa capacité à minimiser les pertes thermiques.

Y2 : Charge de refroidissement (Cooling Load)

La charge de refroidissement est l'énergie nécessaire pour maintenir une température intérieure confortable en été. Elle est influencée par l'exposition solaire et les matériaux de construction.

1-4-Utilité et Objectifs de la Prédiction:

L'objectif de cette prédiction est de fournir des estimations précises des besoins énergétiques en chauffage et en climatisation des bâtiments résidentiels. Grâce aux prédictions de y1 et y2, ce modèle peut aider les architectes, ingénieurs et planificateurs urbains à concevoir des bâtiments plus efficaces d'un point de vue énergétique, répondant aux standards écologiques modernes.

En analysant l'influence de chaque caractéristique sur les besoins énergétiques, il devient possible d'optimiser la conception des bâtiments, de choisir des matériaux adaptés et de planifier des stratégies d'orientation et d'isolation. Ce projet peut donc contribuer à réduire la consommation énergétique des bâtiments, à minimiser les coûts de chauffage et de climatisation, et à abaisser les émissions de carbone, favorisant ainsi une

2-Entraînement d’un modèle

Entraînement d’un modèle pour chaque variable cible .Étant donné que certaines des variables prédictives, notamment X6 (orientation) et X8 (répartition de la surface vitrée .En appliquent plusieurs modèles de machine learning. Chaque modèle sera évalué en utilisant **2-1-Préparation des Données** des métriques appropriées, telles que l'erreur quadratique moyenne (RMSE) . Après avoir comparé les performances des différents modèles, nous sélectionnerons celui qui offre la meilleure précision pour prédire y1 et y2.

:

La première étape consistera à identifier d'éventuelles valeurs manquantes dans le jeu de données :

```
[12]: # Vérifier s'il y a des données manquantes
      data.isna().sum()
```

```
[12]: X1    0
      X2    0
      X3    0
      X4    0
      X5    0
      X6    0
      X7    0
      X8    0
      Y2    0
      dtype: int64
```

Un aperçu global des caractéristiques et de la structure du jeu de données :

```
## Afficher un résumé des données
data.describe(include="all")
```

	X1	X2	X3	X4	X5	X6	X7	X8	Y2
count	768.000000	768.000000	768.000000	768.000000	768.00000	768.000000	768.000000	768.00000	768.000000
mean	0.764167	671.708333	318.500000	176.604167	5.25000	3.500000	0.234375	2.81250	24.587760
std	0.105777	88.086116	43.626481	45.165950	1.75114	1.118763	0.133221	1.55096	9.513306
min	0.620000	514.500000	245.000000	110.250000	3.50000	2.000000	0.000000	0.00000	10.900000
25%	0.682500	606.375000	294.000000	140.875000	3.50000	2.750000	0.100000	1.75000	15.620000
50%	0.750000	673.750000	318.500000	183.750000	5.25000	3.500000	0.250000	3.00000	22.080000
75%	0.830000	741.125000	343.000000	220.500000	7.00000	4.250000	0.400000	4.00000	33.132500
max	0.980000	808.500000	416.500000	220.500000	7.00000	5.000000	0.400000	5.00000	48.030000

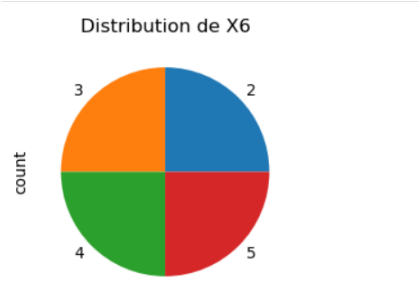
-X7 : (écart-type / moyenne)= 0,57 Indique une grande variabilité relative de la surface vitrée entre les bâtiments.

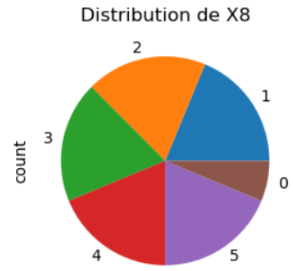
-X8 : (écart-type / moyenne)= 0,55 Montre une forte dispersion dans la distribution de la surface vitrée.

-X3 (Wall Area) : (écart-type / moyenne)= 0,137 Les surfaces murales varient peu en proportion de leur moyenne.

-Les deux variables X1 et X4 sont relativement symétrique (La moyenne et la médiane sont très proches)

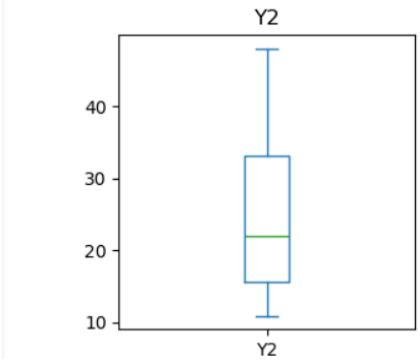
```
for col in qualitative_vars:
    plt.figure(figsize=(3,3))
    data[col].value_counts().plot(kind="pie") # ou : sns.countplot(x=col, data=data)
    plt.title(f"Distribution de {col}")
    plt.show()
```





La variable X6 (Orientation) présente quatre catégories (nord, sud, est, ouest) avec des pourcentages égaux, indiquant une répartition équilibrée entre les orientations. Cela suggère que les modèles que développés ne seront pas biaisés par une orientation dominante.

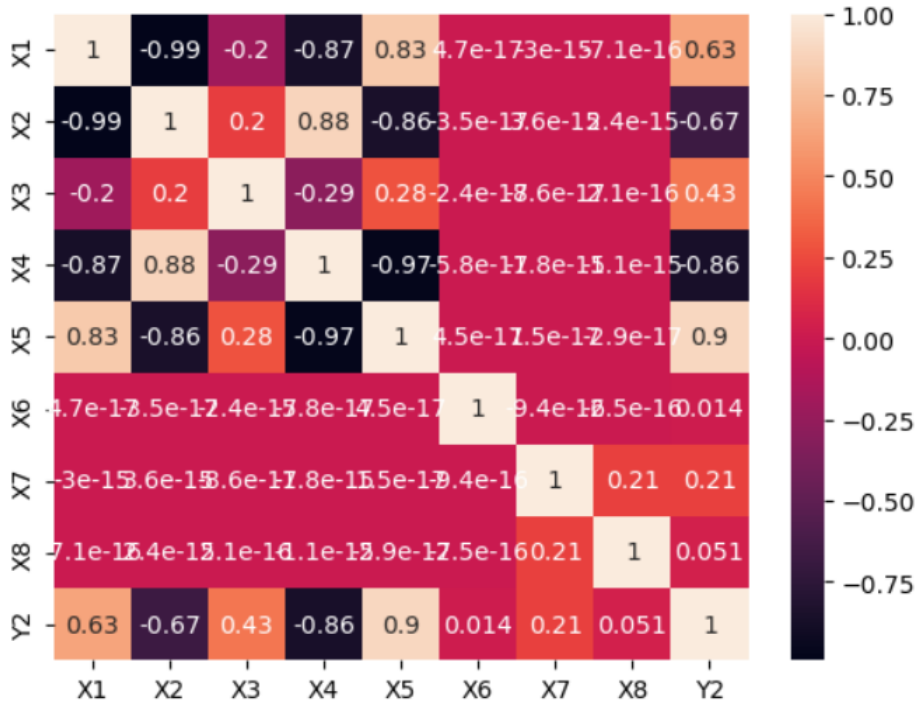
```
[22]: #Décrire les variables quantitative: identification des données aberrantes
vars=['X1', 'X2', 'X3', 'X4', 'X5', 'X7',"Y2"]
for col in vars:
    plt.figure(figsize=(3,3))
    plt.title(f"{col}")
    data[col].plot(kind='box')
    plt.show()
```



-Les deux variables Y1 et Y2 sont relativement asymétrique

```
#afficher la matrice de corrélation
sns.heatmap(data.corr(),annot=True)
##on remarque qu il y a une faible corrélatin entre X6,X8 et Y2
##une corrélation négative enter X2,X4 et Y1
##fort corrélation entre X5,X4 et Y1
```

```
<Axes: >
```



Les variables qualitatives seront d'abord séparées de l'ensemble des variables numériques. Ensuite, un scaler sera appliqué pour les standardiser. Enfin, le jeu de données sera divisé en ensembles d'entraînement et de test.

Ensuite, des modèles de régression tels que Ridge, Lasso et Elastic Net seront entraînés en utilisant des paramètres optimaux.

Theil-Sen Regressor

```
# Définir la grille des paramètres pour Theil-Sen
param_grid_theil_sen = {
    'max_iter': [1000, 2000],  # Nombre d'itérations pour l'optimisation
    'tol': [1e-4, 1e-3],  # Tolérance pour la convergence
    'fit_intercept': [True, False]  # Si le modèle linéaire doit inclure un intercept
}

# Créer et entraîner le modèle Theil-Sen avec GridSearch
theil_sen = TheilSenRegressor()
grid_theil_sen = GridSearchCV(estimator=theil_sen, param_grid=param_grid_theil_sen, scoring='r2', cv=5)
grid_theil_sen.fit(x_train, y_train)

# Afficher les meilleurs paramètres de Theil-Sen
print("\nMeilleurs paramètres pour Theil-Sen Regressor :")
print(grid_theil_sen.best_params_)

# Évaluer le modèle avec les meilleurs paramètres
print("\nTheil-Sen Regressor:")
model_theil_sen = grid_theil_sen.best_estimator_
evaluate_model(model_theil_sen, x_train, y_train, x_test, y_test)
```

Pour Y1 :

model	RANSAC Regressor	Theil-Sen Regressor	Ridge	Lasso	ElasticNet
MAE	2.348536	2.299419	2.340451	2.341232	2.337733
MSE	10.525493	9.796672	10.047241	10.044940	10.056338
R^2	0.904075	0.910717	0.908433	0.908454	0.908350

Pour Y2 :

model	RANSAC Regressor	Theil-Sen Regressor	Ridge	Lasso	ElasticNet
MAE	2.246098	2.406592	2.314548	2.314407	2.308553
MSE	11.656220	11.965231	11.556047	11.553883	11.556186
R^2	0.882036	0.878909	0.883050	0.883072	0.883049

3-Déploiement d'une machine learning

la création d'une API avec Flask nécessite généralement un fichier Python pour configurer et démarrer l'application. Ce fichier contient le code nécessaire pour définir les routes de l'API, les fonctions de traitement des requêtes, et les réponses de l'API.

Importation des Bibliothèques :

On commence par importer Flask et d'autres modules nécessaires :

```
from flask import Flask, request, jsonify, render_template
import joblib
import numpy as np
import pandas as pd
```

Création de l'Application et importation de modèles et des scaler

```
app = Flask(__name__)
# Load the Random Forest model
model1 = joblib.load('model1.pkl')
model2 = joblib.load('model2.pkl')

scaler1 = joblib.load('scaler1.pkl')  # Chargez le scaler pour le modèle 1
scaler2 = joblib.load('scaler2.pkl')  # Chargez le scaler pour le modèle 2
```

Définition des Routes :Les routes permettent de définir les différentes pages de l'application. Et définitions des fonctions par exemple :

-Fonction pour afficher le contenue du fichier html hoom .html

```
@app.route('/')
def home():
    return render_template('hoom.html')
```

-Fonction pour faire la prédiction

```python
def predict():
    # Récupérer les données JSON de la requête
    data = request.get_json(force=True)

    # Vérifier quel modèle utiliser
    model_type = data.get('model')
    data.pop('model', None)  # Supprimer le champ 'model' des données

    # Définir les variables binaires (pour les deux modèles)
    binary_variables = ["X6", "X8"]

    # Créer un DataFrame pour les variables non binaires
    non_binary_variables = {key: data[key] for key in data.keys() if key not in binary_variables}
    non_binary_df = pd.DataFrame(non_binary_variables, index=[0])

    # Faire la prédiction en fonction du modèle spécifié
    if model_type == 1:
        # Standardiser les variables non binaires
        dfstand = scaler1.transform(non_binary_df)
        dfstand = pd.DataFrame(dfstand, columns=non_binary_df.columns)

        # Ajouter les variables binaires au DataFrame standardisé
        binary_df = pd.DataFrame(data, index=[0])[binary_variables]
        Xs_test_final = pd.concat([dfstand, binary_df], axis=1)

        # Réorganiser les colonnes pour suivre l'ordre original
        column_order = ['X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X7', 'X8']
        Xs = Xs_test_final[column_order]

        prediction1 = model1.predict(Xs)
        return jsonify({'prediction1': prediction1[0]})
```

En plus du fichier python, il y’a d'autres fichiers et dossiers dans la structure du projet:

-Requirement.txt : Ce fichier liste toutes les dépendances nécessaires à l'application.

requirements.txt

```
Flask
joblib
numpy
scikit-learn
pandas
```

-Fichier html : Un fichier HTML est utilisé avec Flask pour structurer et afficher l'interface utilisateur de l'application Web. Flask, en tant que framework backend, gère la logique et le traitement des données, mais il ne fournit pas directement l'interface graphique. Le fichier HTML sert donc à créer cette interface visuelle que l'utilisateur voit et utilise dans le navigateur.

-Un fichier Dockerfile :est essentiel pour déployer une application Flask dans un conteneur Docker. Il contient une série d'instructions qui définissent comment construire l'image Docker de l'application, en installant les dépendances et en configurant l'environnement pour que Flask fonctionne correctement. Cela permet de rendre l'application portable, facile à déployer et compatible avec divers environnements (local, cloud, etc.).

```
# Utiliser une image Python officielle
FROM python:3.13

# Définir le répertoire de travail
WORKDIR /app

# Copier le fichier requirements.txt et installer les dépendances
COPY requirements.txt requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Copier le code de l'application et le modèle
COPY . .

# Exposer le port de l'application
EXPOSE 5000

# Commande pour exécuter l'application
CMD ["python", "app.py"]
```

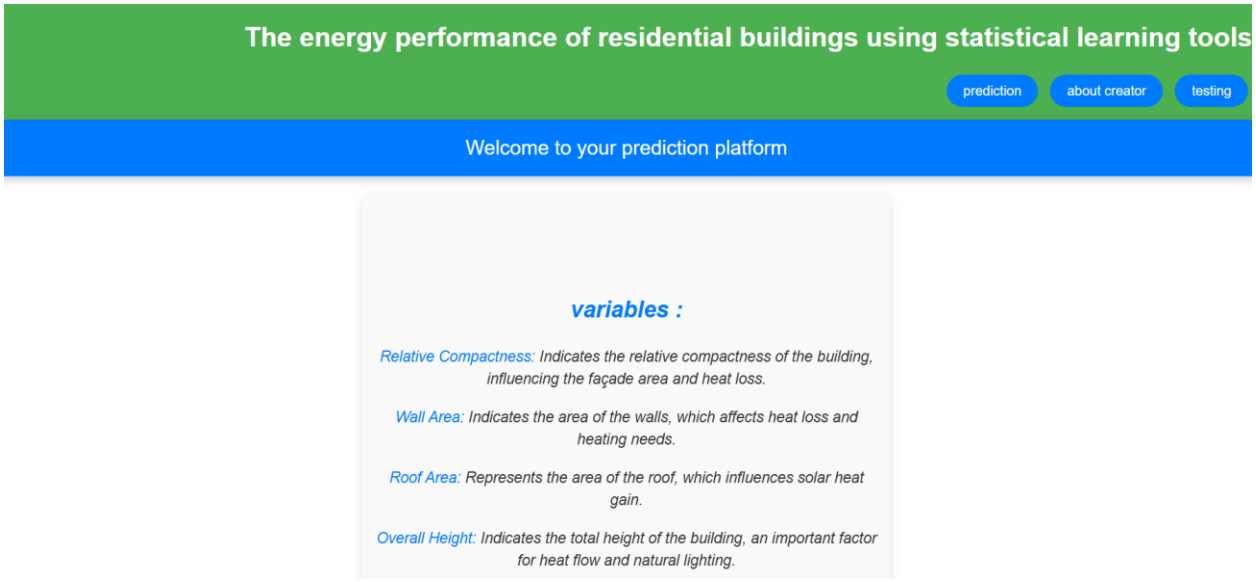
4-Présentation de l'Interface Utilisateur

-Lien vers API :

https://abdo-486638708818.europe-southwest1.run.app/

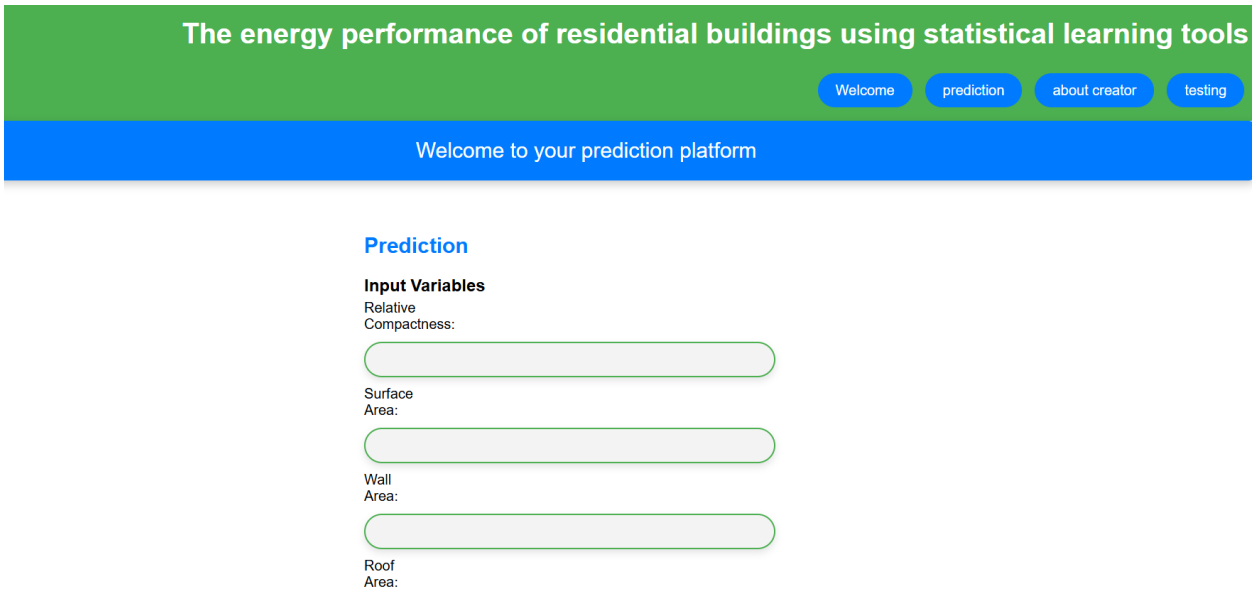
-Page d'Accueil

La page d'accueil de l'interface utilisateur offre un aperçu général des différentes variables utilisées dans le modèle ainsi que des variables cibles.



une page de prédiction :

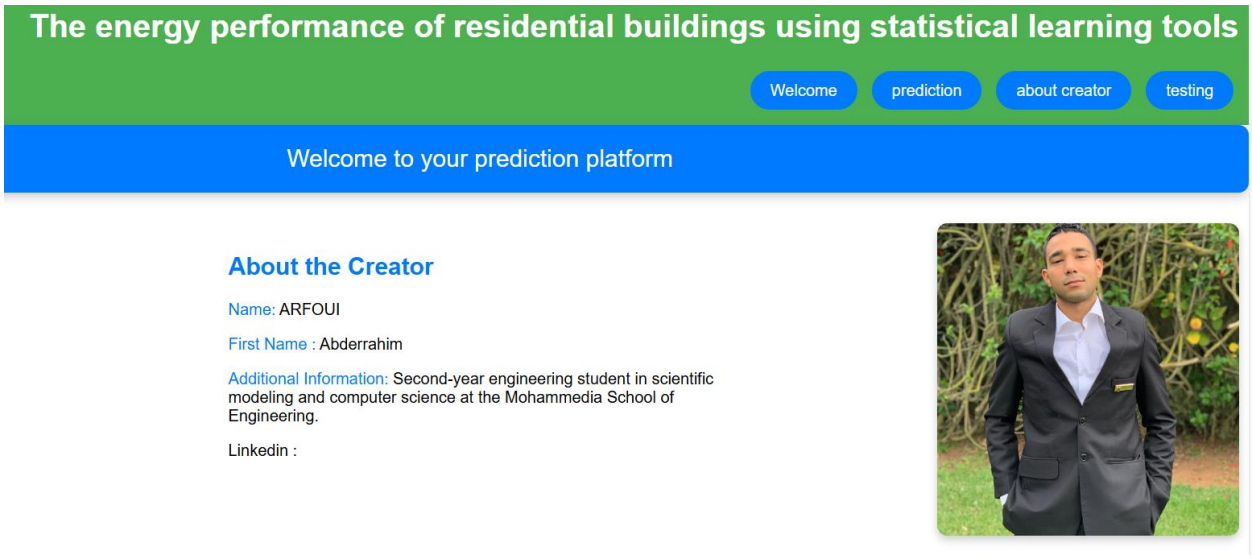
dans laquelle l'utilisateur peut saisir les valeurs des variables pour obtenir des prédictions sur les cibles (Y1 et Y2). Cette organisation permet de guider efficacement les utilisateurs, depuis la compréhension des variables jusqu'à la génération de prédictions via le modèle.



Une page de test :

Permettant de vérifier le bon fonctionnement de la fonctionnalité de prédiction. Cette page offre aux utilisateurs un espace pour soumettre des valeurs d'exemple et évaluer la précision des prédictions du modèle, garantissant ainsi que le système fonctionne comme attendu avant une utilisation sur des données réelles.

Une page d'informations sur le créateur :



Conclusion du Projet

Ce projet a consisté à développer une application web avec une API dédiée à la prédiction des charges énergétiques des bâtiments, spécifiquement la charge de chauffage (Y1) et la charge de refroidissement (Y2). Le processus a impliqué plusieurs étapes clés:

1. **Analyse des données** : Nous avons utilisé un ensemble de données comprenant 8 variables prédictives telles que la compacité relative, la surface totale, la surface des murs, la

surface du toit, la hauteur totale, l'orientation, la surface vitrée, et la répartition de la surface vitrée.

2. **Développement de l'API** : Une API a été créée en utilisant Flask pour faciliter les prédictions en temps réel. L'API accepte les entrées des utilisateurs sous la forme de paramètres des bâtiments et retourne les prédictions des charges énergétiques.
3. **Interface utilisateur** : Une interface web simple permet aux utilisateurs de saisir les paramètres des bâtiments et d'obtenir instantanément les prédictions. L'interface inclut également une vérification de l'état du service.
4. **Implémentation et déploiement** : L'API a été testée et déployée, permettant un accès facile pour l'évaluation des performances énergétiques. Cette solution est conçue pour être extensible et adaptable aux besoins futurs.