Seattle

ANTICIPEZ LES BESOINS EN CONSOMMATION DE BÂTIMENTS

Presentation - 2024

Source database
data.seattle.gov

Vision & Mission

Vision

Vers une Ville Durable : Réduire les Émissions de Carbone des Bâtiments Non Résidentiels

La consommation énergétique et les émissions de carbone provenant des bâtiments non résidentiels sont des pièces maîtresses de notre empreinte carbone collective. Pour concrétiser notre vision de durabilité, il est impératif d'adopter des approches novatrices et proactives visant à réduire ces émissions.

Mission

Réduire les Émissions de CO2 dans les Bâtiments Non Résidentiels de Seattle : Une Approche Analytique et Prédictive

Notre mission au sein de l'équipe de la ville de Seattle est guidée par la volonté de faire de notre ville une référence en matière de durabilité urbaine. Suite à des relevés minutieux effectués en 2016 par nos agents municipaux, nous disposons de données cruciales sur la consommation énergétique et les émissions de CO2 des bâtiments non destinés à l'habitation. Cependant, ces relevés coûtent cher, et notre objectif est d'optimiser l'utilisation de ces données en tentant de prédire les émissions pour les bâtiments non encore évalués. Cette prédiction reposera sur les caractéristiques structurelles des bâtiments, des variables tel que ENERGYSTARScore, emission G.E.S...

Sommaire

01 Préparation & présentation du dataset

Une analyse exploratoire approfondie des données de 2016 est en cours. Nous cherchons à identifier les tendances, les corrélations, et les caractéristiques les plus influentes sur les émissions de CO2 et la consommation d'énergie.

02 Modélisation Prédictive

On se consacrera à tester différents modèles de prédiction pour maximiser la précision de nos estimations. Nous explorerons des algorithmes de machine learning tels que ElasticNet, SVM, GradientBoosting, RandomForest, tout en intégrant les données des bâtiments.

03 Prédiction des indicateurs consommation d'energy et émission

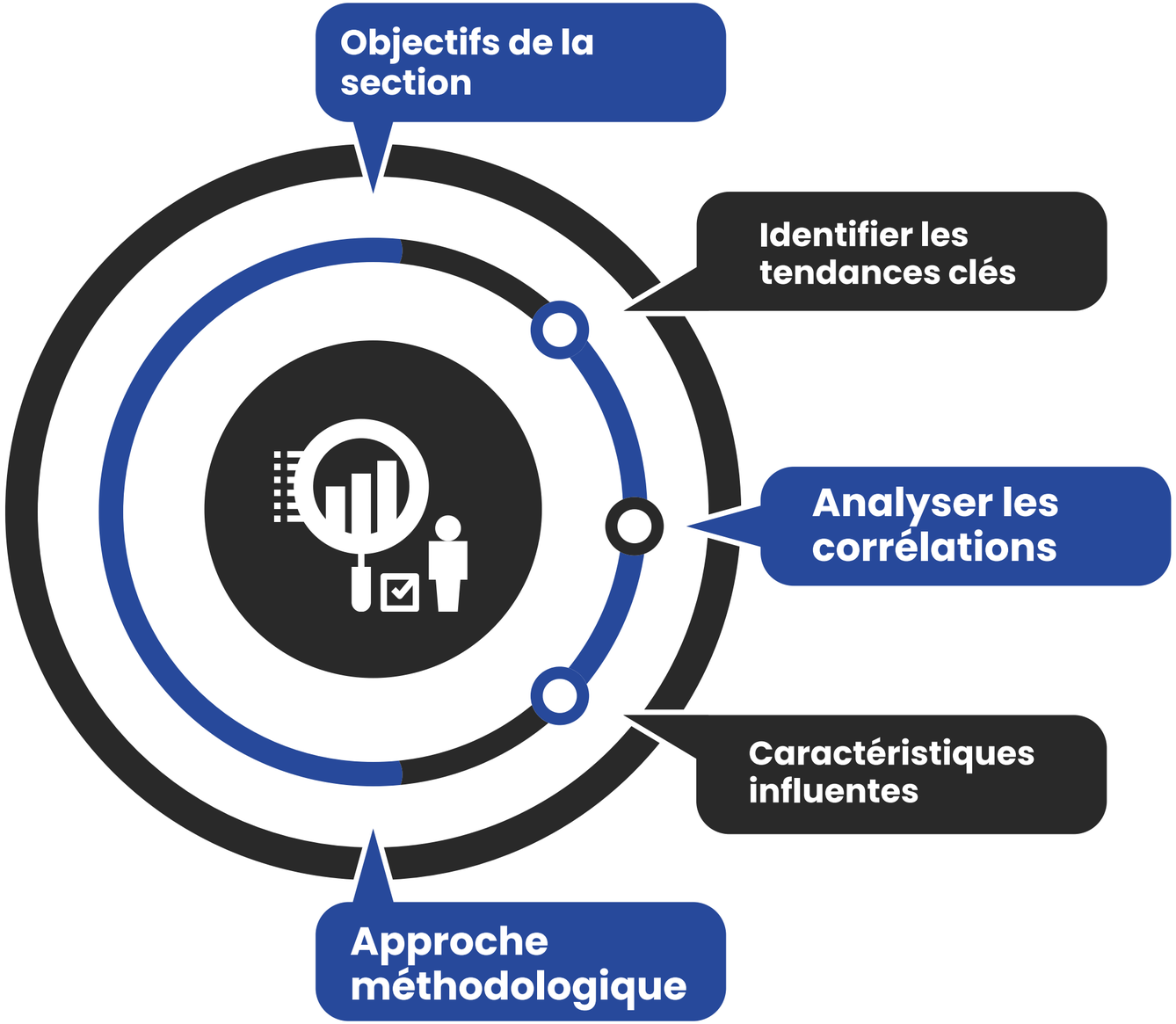
1) Prédiction de la consommation d'énergie
2) Prédiction des émissions

04 Prédiction sans la variable ENERGY STAR Score

Nous évaluerons l'intérêt de l'ENERGY STAR Score, un indicateur de performance énergétique, pour améliorer la prédiction des émissions de CO2. Nous explorerons son utilité potentielle en l'excluant de notre modélisation. Si son exclusion se traduit par une diminution significative de la précision, cela pourrait justifier l'investissement nécessaire pour son calcul.

01 : Préparation & présentation du dataset

L'analyse exploratoire sera menée de manière systématique, en utilisant des outils statistiques et des visualisations pour mettre en lumière les insights clés. Les résultats de cette étape fourniront la base nécessaire pour orienter nos choix de modèles prédictifs dans la section suivante.



DATASET

Date: 01-2024

GÉOGRAPHIQUES

- Longitude
- Latitude
- Neighborhood

CARACTÉRISTIQUE

- Nombre de bâtiments
- Nombre d'étages
- Année de construction
- propertyGFATotal (Surface)

USAGE

- BuildingType (Residential, non résidential, multi family, campus)
- primarypropertyType (hotel, University, laboratory, restaurant ...)

CONSOMMATION & EMISSION

- SiteEUIWN(kBtu/sf)
- TotalGHGEmissions

MODIFICATION & CRÉATION VARIABLES

- Ancienneté
- BuildingAreapercentage
- ParkingAreaPercentage
- AverageGFAperBui1ding
- AverageGFAperF100r

NOUVELLES VARIABLE EXPLICATION

- **BuildingAreaPercentage :**
  1. **Calcul** : La variable **BuildingAreaPercentage** est obtenue en divisant la superficie de construction du bâtiment (**PropertyGFABuilding**(s)) par la superficie totale du bâtiment (**PropertyGFATotal**).
- **ParkingAreaPercentage :**
  1. **Calcul :** La variable **ParkingAreaPercentage** est obtenue en divisant la superficie de stationnement du bâtiment (**PropertyGFAParking**) par la superficie totale du bâtiment (**PropertyGFATotal**).
- **Ancienneté** :
  1. **Calcul** : La variable **Ancienneté** est créée en soustrayant l'année de construction du bâtiment (**YearBuilt**) de l'année de relevé des données (**DataYear**). Cela donne l'âge du bâtiment au moment du relevé.
- **AverageGFAperBuilding :**
  1. **Calcul :** La variable **AverageGFAperBuilding** est obtenue en divisant la superficie totale du bâtiment (**PropertyGFATotal**) par le nombre total de bâtiments (**NumberofBuildings**). Cela donne la surface moyenne par bâtiment.
- **AverageGFAperFloor :**
  1. **Calcul :** La variable **AverageGFAperFloor** est obtenue en multipliant la superficie totale du bâtiment (**PropertyGFATotal**) par le nombre d'étages du bâtiment (**NumberofFloors**). Cela donne la superficie totale par étage.

DATASET

Informations générales :

- **Nom de la base de données** : energy_benchmarking
- **Nombre d'entrées** : 3376
- **Nombre de colonnes** : 46

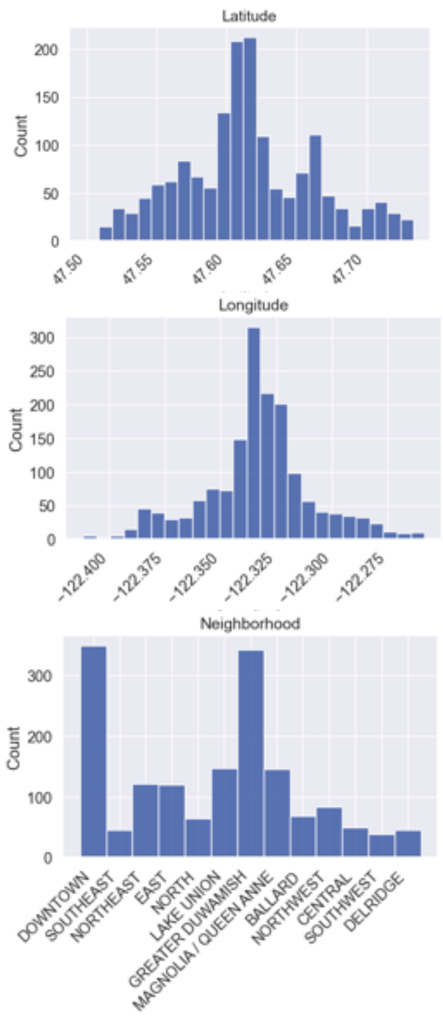
Remarques importantes :

- **Données manquantes :** Certaines colonnes ont des valeurs manquantes (par exemple, ZipCode, NumberofBuildings, LargestPropertyUseType, etc.).
- **Variables énergétiques :** Plusieurs colonnes concernent la consommation d'énergie, les émissions de gaz à effet de serre, et les scores ENERGY STAR.
- **Données par défaut :** La colonne DefaultData indique si les données sont par défaut.
- **Commentaires :** La colonne Comments ne contient aucune donnée (toutes les entrées sont nulles).

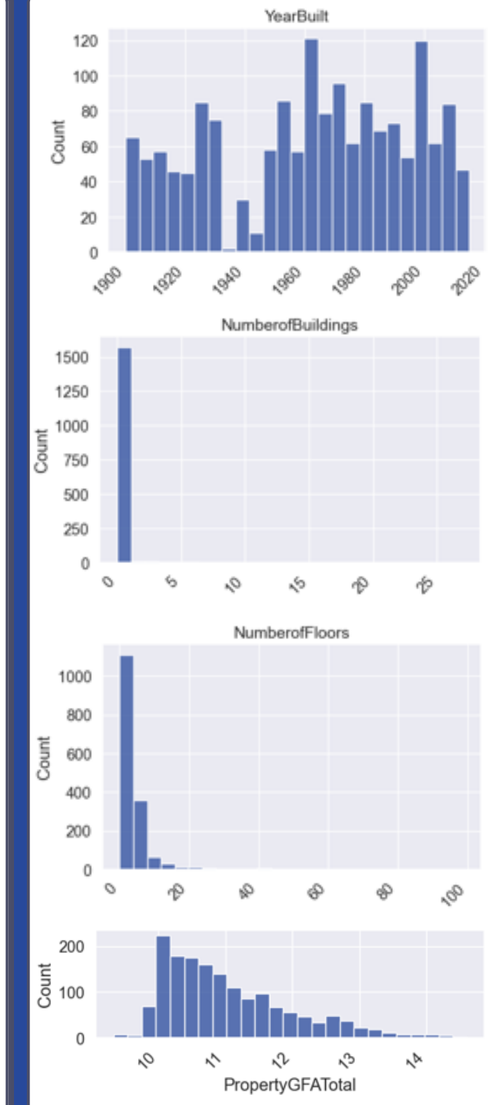
RÉPARTITION EMPIRIQUE DES DONNÉES

Date: 01-2024

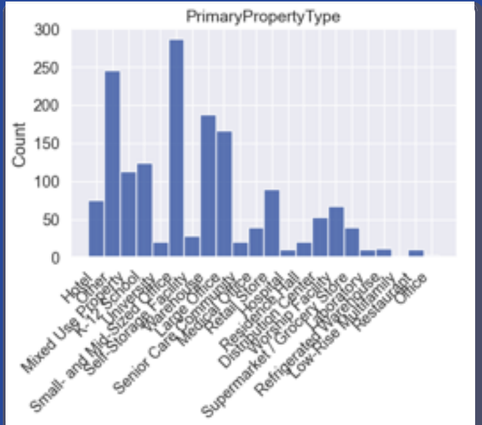
GÉOGRAPHIQUES



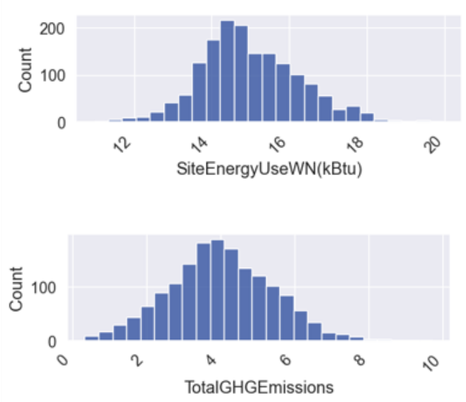
CARACTÉRISTIQUE



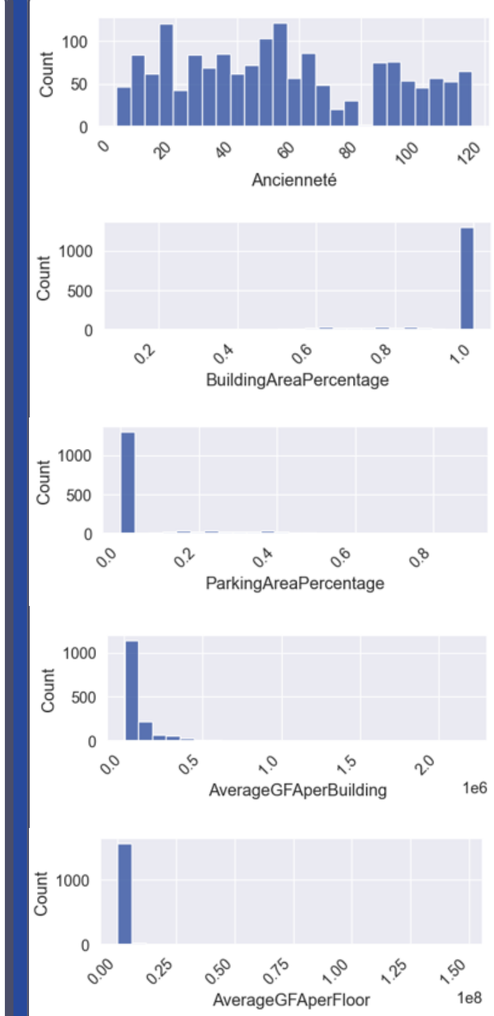
USAGE



CONSOMMATION & EMISSION



MODIFICATION & CRÉATION VARIABLES



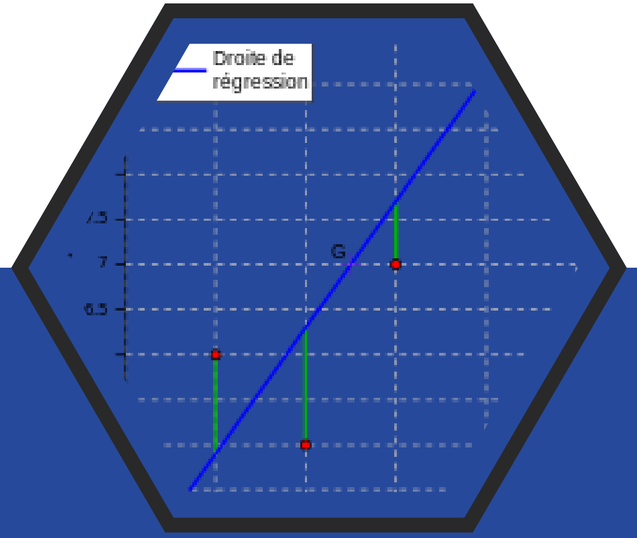
02 Modélisation préditive

1. Prédiction de la consommations et des Emissions
   - Comparaison sur 5 modèles de machine learning différent, optimisés par validation croisée.
   - Données standardisées, normalisé
   - Score : 4 métriques
2. CHoix du meilleur model avec courbe d'apprentissage
3. Importer la variable **ENERGYSTARScore**

Models utilisés

```python
# Liste des modèles à tester
models = {
    "KNN": KNeighborsRegressor(),
    "ElasticNet": ElasticNet(),
    "GradientBoosting": GradientBoostingRegressor(),
    "SVR": SVR(),
    "RandomForest": RandomForestRegressor()
}

# Grilles de recherche pour chaque modèle
grid_configs = {
    "KNN": {"regressor__n_neighbors": range(2, 15)},
    "ElasticNet": {"regressor__alpha": np.logspace(0.00001, 10, 30), "regressor__l1_ratio": [0.1, 0.3, 0.5, 0.7, 0.9]},
    "GradientBoosting": {"regressor__n_estimators": np.linspace(10, 400, 30, dtype=int), "regressor__learning_rate": [1,
    "SVR": {"regressor__C": np.logspace(-2, 3, 6)},
    "RandomForest": {"regressor__n_estimators": np.linspace(10, 200, 20,dtype=int), "regressor__min_samples_leaf": [1, 4]
}

# Liste des préprocesseurs pour tester différentes combinaisons
preprocessor_configs = [
    ("RobustScaler + OneHotEncoder", RobustScaler(), OneHotEncoder(handle_unknown='ignore')),
    ("StandardScaler + OneHotEncoder", StandardScaler(), OneHotEncoder(handle_unknown='ignore')),
    ("MinMaxScaler + OneHotEncoder", MinMaxScaler(), OneHotEncoder(handle_unknown='ignore')),
    ("PolynomialFeatures + OneHotEncoder", PolynomialFeatures(), OneHotEncoder(handle_unknown='ignore')),
    ("RobustScaler + TargetEncoder", RobustScaler(), TargetEncoder(cols=categorical_features)),
    ("StandardScaler + TargetEncoder", StandardScaler(), TargetEncoder(cols=categorical_features)),
    ("MinMaxScaler + TargetEncoder", MinMaxScaler(), TargetEncoder(cols=categorical_features)),
    ("PolynomialFeatures + TargetEncoder", PolynomialFeatures(), TargetEncoder(cols=categorical_features)),
]
```

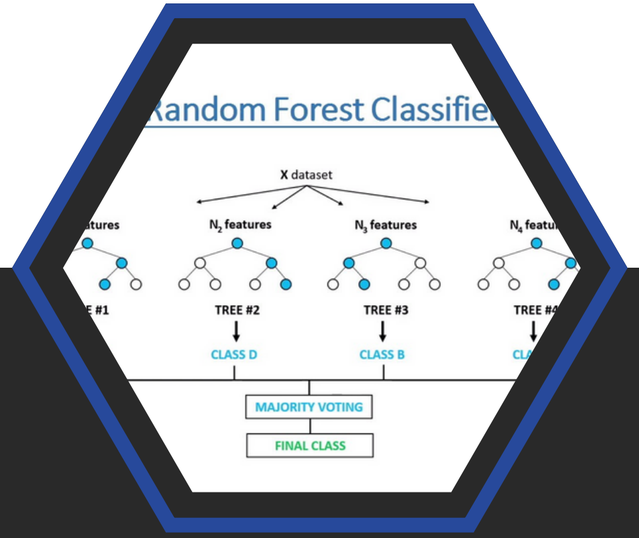
Modèles utilisés



Régression linéaire

ElasticNet combine à la fois la pénalité L1 (lasso) et la pénalité L2 (ridge) dans le but de bénéficier des avantages des deux approches. Il est souvent utilisé dans le contexte de la régression linéaire pour éviter le surajustement (overfitting) du modèle aux données d'entraînement.

Les paramètres principaux de **ElasticNet** sont alpha et l1_ratio



Ensemblistes

RandomForestRegressor est un modèle ensembliste appartenant à la famille des méthodes d'ensemble, qui combine plusieurs modèles pour améliorer les performances prédictives.

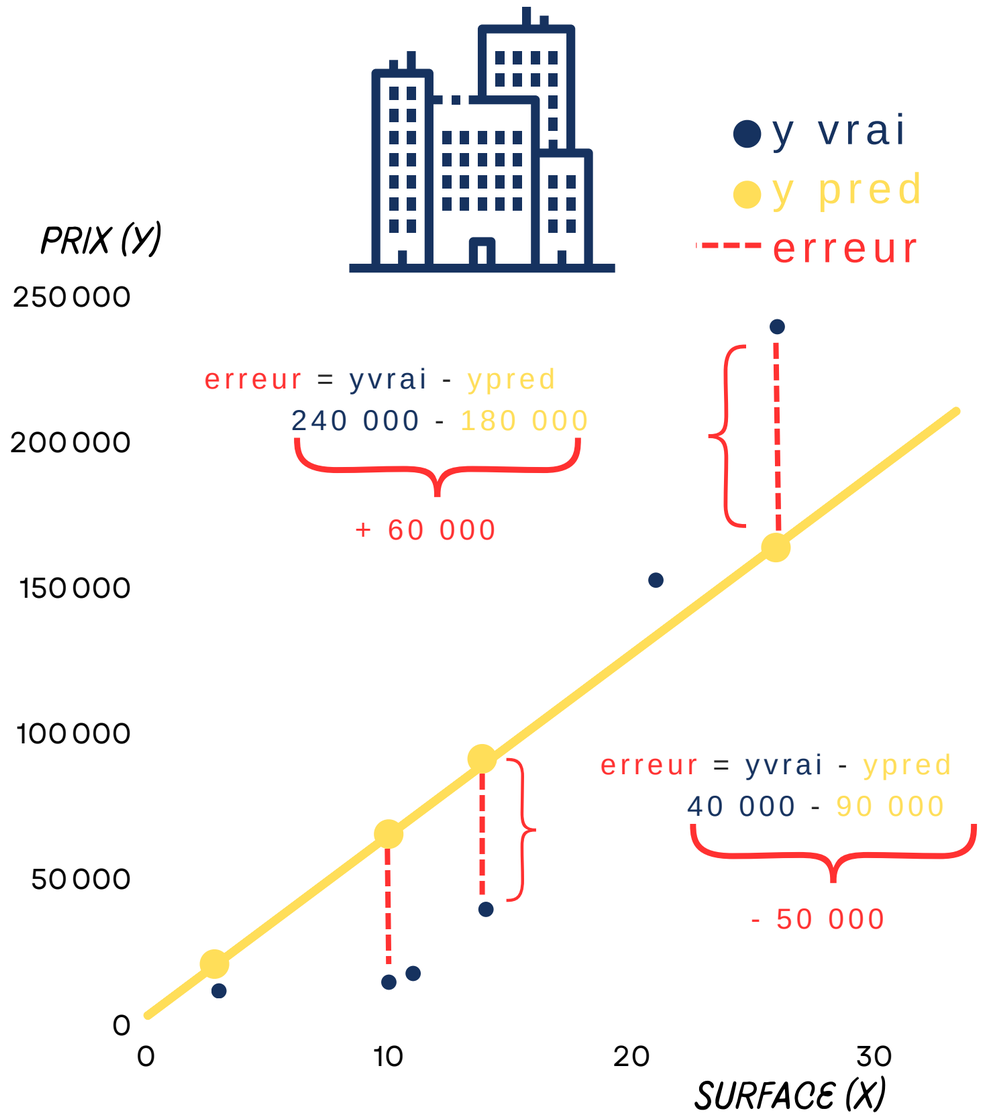
Le RandomForestRegressor a plusieurs paramètres, tels que le nombre d'arbres dans la forêt, la profondeur maximale des arbres, et d'autres paramètres de division qui peuvent être ajustés pour optimiser les performances du modèle.

Support vector machine

Les machines à vecteurs de support pour la régression (**SVR**) sont une variante des machines à vecteurs de support (**SVM**) utilisées dans le contexte de la régression plutôt que de la classification.

C dans les **SVR** contrôle la trade-off entre la précision du modèle sur les données d'entraînement et sa capacité à généraliser à de nouvelles données. Il est important de régler soigneusement ce paramètre pour obtenir un équilibre optimal entre surajustement et sous-ajustement, ce qui permet d'obtenir des performances maximales sur des données non vues.

Métriques MSE, MAE



Travailler avec des erreurs soit positives soit négatives n'est pas pratique.

Pour corriger ce probleme nous avons deux options:

- **Prendre la valeur absolue de chaque erreur**

$$\text{erreur absolue} = |\ \text{yvrai} - \text{ypred}\ |$$

$$\underbrace{40\,000 - 90\,000}_{50\,000}$$

- **calculer le carré de chaque erreur**

$$\text{erreur quadratique} = (\ \text{yvrai} - \text{ypred}\ )^2$$

$$\underbrace{40\,000 - 90\,000}_{2\,500\,000\,000}$$

La moyenne des erreurs

- **Erreur quadratique moyenne (<u>Mean Squared Error</u>) :**

$$\text{MSE} = \frac{1}{m} \sum (\ \text{yvrai} - \text{ypred}\ )^2$$

Si on accorde une grande importance aux grandes erreurs.

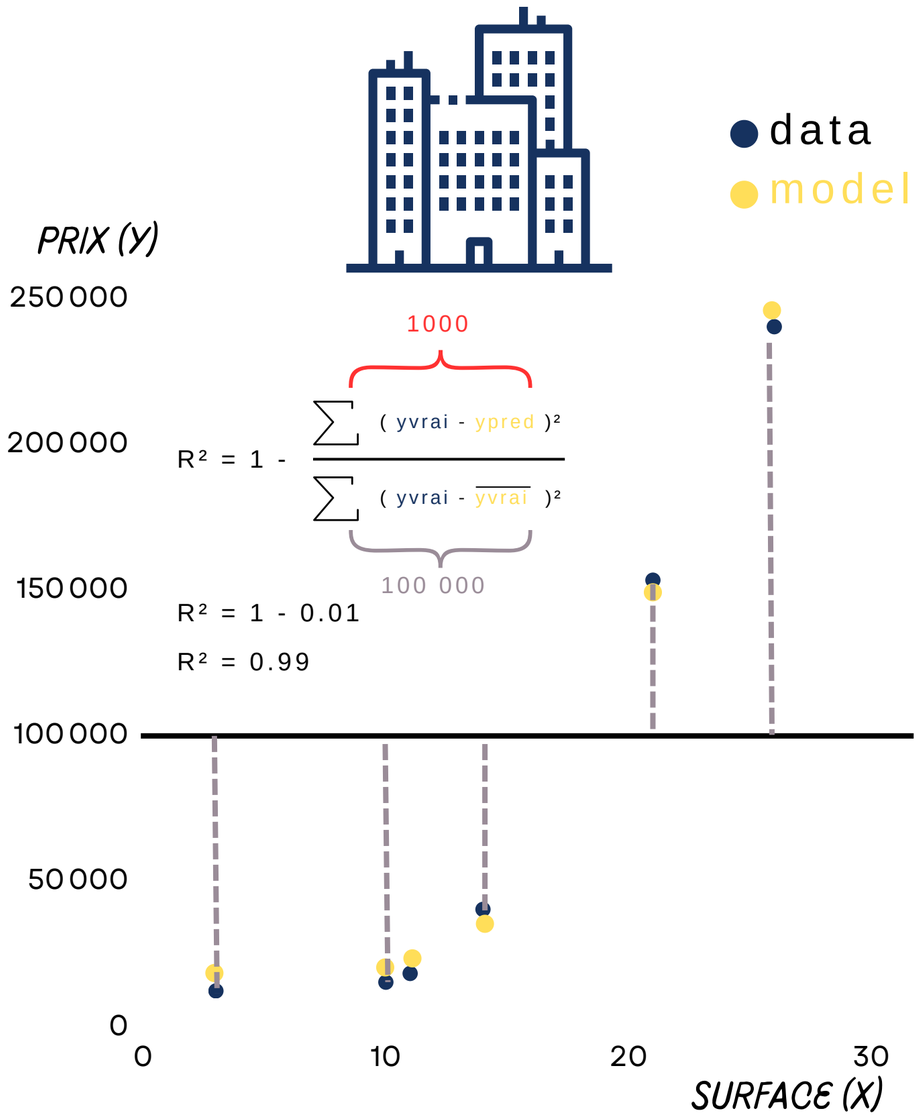
- **Erreur absolue moyenne (<u>Mean Absolute Error</u>) :**

$$\text{MAE} = \frac{1}{m} \sum |\ \text{yvrai} - \text{ypred}\ |$$

$$\sum (\ \text{erreurs})^2 \neq \left(\sum \text{erreurs}\right)^2$$

L'importance d'une erreur est linéaire avec son amplitude. Si le Dataset contient des outliers.

Métriques R²



Erreur quadratique

$$R^2 = 1 - \frac{\boxed{\sum (yvrai - ypred)^2}}{\boxed{\sum (yvrai - \overline{yvrai})^2}}$$

Variance

moyenne des yvrai

Evalue la performance du modèle par rapport au niveau de variation présent dans les données.

$$\boxed{R^2} = 1 - \frac{\sum (yvrai - ypred)^2}{\sum (yvrai - \overline{yvrai})^2}$$

1

$$\frac{\text{Erreur}}{\text{Variance}}$$

0

Erreur << Variance

03 Prédiction de la consommation énergétique

| Les modèles ayant un score de test supérieur à 0.60 sont : |

```
Nom du modèle : ElasticNet - PolynomialFeatures + OneHotEncoder
Configuration : {'regressor__alpha': 17433398.957881466, 'regressor__l1_ratio': 0.9}
Test Score : 0.6851488502412671
Nom du modèle : ElasticNet - PolynomialFeatures + TargetEncoder
Configuration : {'regressor__alpha': 3562276.1744398694, 'regressor__l1_ratio': 0.1}
Test Score : 0.6583297564896539
Nom du modèle : GradientBoosting - PolynomialFeatures + TargetEncoder
Configuration : {'regressor__learning_rate': 0.05, 'regressor__n_estimators': 292}
Test Score : 0.6718164481877327
Nom du modèle : RandomForest - PolynomialFeatures + OneHotEncoder
Configuration : {'regressor__min_samples_leaf': 1, 'regressor__n_estimators': 40}
Test Score : 0.6597473023090468
Nom du modèle : RandomForest - RobustScaler + TargetEncoder
Configuration : {'regressor__min_samples_leaf': 1, 'regressor__n_estimators': 30}
Test Score : 0.689009616529671
Nom du modèle : RandomForest - StandardScaler + TargetEncoder
Configuration : {'regressor__min_samples_leaf': 1, 'regressor__n_estimators': 30}
Test Score : 0.6811161347389083
Nom du modèle : RandomForest - MinMaxScaler + TargetEncoder
Configuration : {'regressor__min_samples_leaf': 1, 'regressor__n_estimators': 30}
Test Score : 0.6884698335197708
Nom du modèle : RandomForest - PolynomialFeatures + TargetEncoder
Configuration : {'regressor__min_samples_leaf': 1, 'regressor__n_estimators': 60}
Test Score : 0.6858077992713902
```

Prédiction de la consommation énergétique

| Model final : |

```
Nom du modèle : GradientBoosting - PolynomialFeatures + TargetEncoder
Configuration : {'regressor__learning_rate': 0.05, 'regressor__n_estimators': 292}
Test Score : 0.6718164481877327
```

```python
# Diviser les données
X = energy_benchmarking_ml.dropna().drop(columns=['SiteEnergyUseWN(kBtu)'])
y = energy_benchmarking_ml.dropna()['SiteEnergyUseWN(kBtu)']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)

best_model = GradientBoostingRegressor(learning_rate=0.05, n_estimators=292)

pipeline = Pipeline(
    steps=[("preprocessor", preprocessor), ("regressor", best_model)]
)

pipeline
```

```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num', PolynomialFeatures(),
                                                  ['Ancienneté',
                                                   'BuildingAreaPercentage',
                                                   'ParkingAreaPercentage',
                                                   'NumberofBuildings',
                                                   'NumberofFloors',
                                                   'PropertyGFATotal',
                                                   'ENERGYSTARScore']),
                                                 ('cat',
                                                  TargetEncoder(cols=['Neighborhood',
                                                                      'PrimaryPropertyType',
                                                                      'BuildingType',
                                                                      'LargestPropertyUseType']),
                                                  ['Neighborhood',
                                                   'PrimaryPropertyType',
                                                   'BuildingType',
                                                   'LargestPropertyUseType'])])),
                ('regressor',
                 GradientBoostingRegressor(learning_rate=0.05,
                                           n_estimators=292))])
```

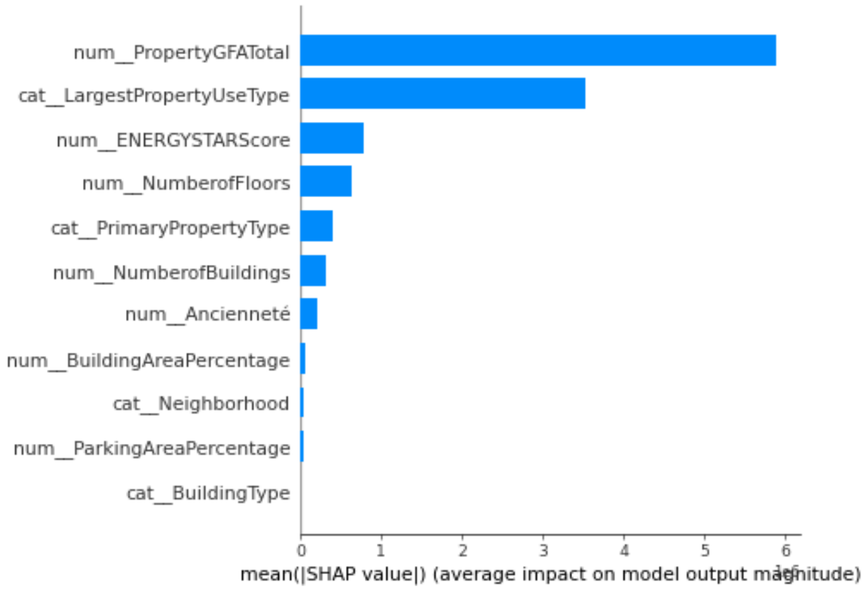
```python
# Entraîner la pipeline sur l'ensemble d'entraînement
pipeline.fit(X_train, y_train)

# Prédire avec le modèle sur l'ensemble de test
y_pred = pipeline.predict(X_test)
```

```python
print("R² (Test Set) :", round(r2_score(y_test, y_pred),2))
```

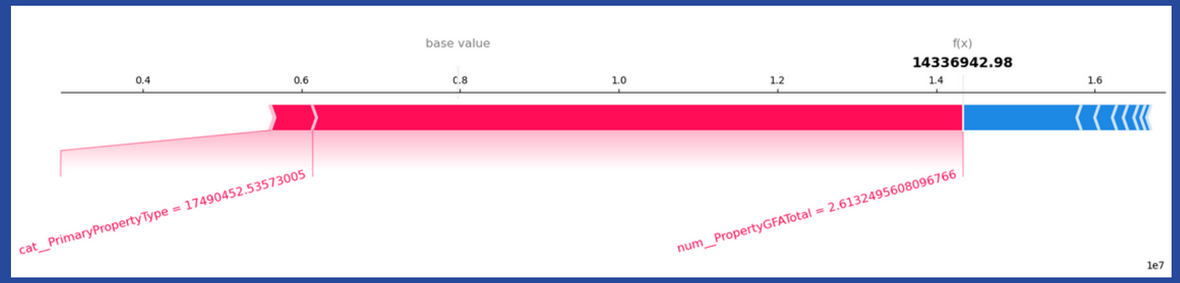
```
R² (Test Set) : 0.9
```

Importance des variables (SHAP)

Variables les plus importante dans notre dataset



Impact des variables sur un individu



higher ⇄ lower

03 Prédiction des emission de G.E.S

| Les modèles ayant un score de test supérieur à 0.50 sont : |

```
Nom du modèle : GradientBoosting - RobustScaler + TargetEncoder
Configuration : {'regressor__learning_rate': 0.1, 'regressor__n_estimators': 225}
Test Score : 0.5285747421339091
Nom du modèle : GradientBoosting - StandardScaler + TargetEncoder
Configuration : {'regressor__learning_rate': 0.25, 'regressor__n_estimators': 319}
Test Score : 0.5091402255700304
Nom du modèle : GradientBoosting - MinMaxScaler + TargetEncoder
Configuration : {'regressor__learning_rate': 0.1, 'regressor__n_estimators': 238}
Test Score : 0.5229046715725287
Nom du modèle : GradientBoosting - PolynomialFeatures + TargetEncoder
Configuration : {'regressor__learning_rate': 0.25, 'regressor__n_estimators': 171}
Test Score : 0.5171816129393968
```

Prédiction de la consommation énergétique

| Model final : |

```
Nom du modèle : GradientBoosting - RobustScaler + TargetEncoder
Configuration : {'regressor__learning_rate': 0.1, 'regressor__n_estimators': 225}
Test Score : 0.5285747421339091
```

```python
# Diviser les données
X = energy_benchmarking_ml.dropna().drop(columns=['TotalGHGEmissions'])
y = energy_benchmarking_ml.dropna()['TotalGHGEmissions']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=40)

best_model = GradientBoostingRegressor(learning_rate=0.1, n_estimators=225, random_state=40)

pipeline = Pipeline(
    steps=[("preprocessor", preprocessor), ("regressor", best_model)]
)

pipeline
```

```
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num', RobustScaler(),
                                                  ['Ancienneté',
                                                   'BuildingAreaPercentage',
                                                   'ParkingAreaPercentage',
                                                   'NumberofBuildings',
                                                   'NumberofFloors',
                                                   'PropertyGFATotal',
                                                   'ENERGYSTARScore']),
                                                 ('cat',
                                                  TargetEncoder(cols=['Neighborhood',
                                                                      'PrimaryPropertyType',
                                                                      'BuildingType',
                                                                      'LargestPropertyUseType']),
                                                  ['Neighborhood',
                                                   'PrimaryPropertyType',
                                                   'BuildingType',
                                                   'LargestPropertyUseType'])])),
                ('regressor',
                 GradientBoostingRegressor(n_estimators=225, random_state=40))])
```

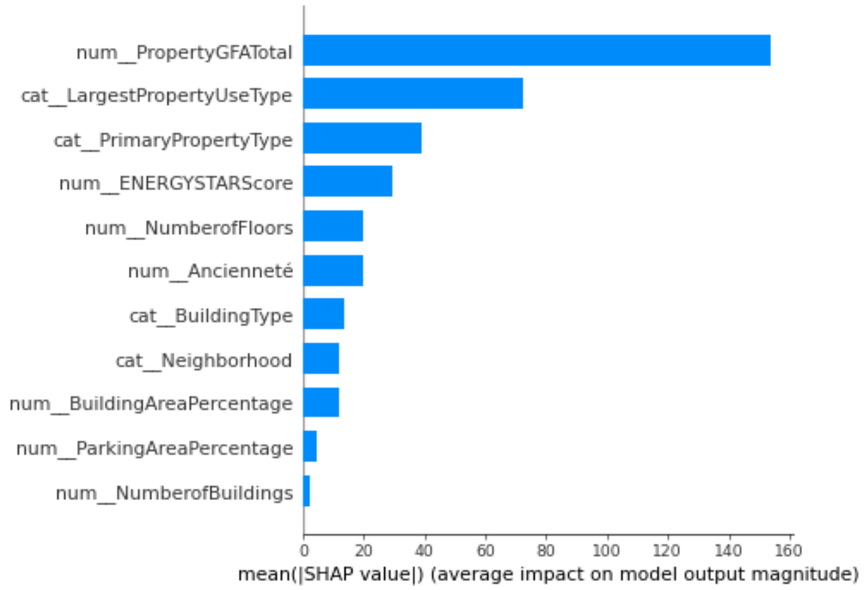
```python
# Entraîner la pipeline sur l'ensemble d'entraînement
pipeline.fit(X_train, y_train)

# Prédire avec le modèle sur l'ensemble de test
y_pred = pipeline.predict(X_test)
```

```python
print("R² (Test Set) :", r2_score(y_test, y_pred))
print("MSE (Test Set) :" , mean_squared_error(y_test, y_pred,squared=False).round(2))
print("RMSE (Test Set) :", mean_squared_error(y_test, y_pred).round(2))
print("MAE (Test Set) :", mean_absolute_error(y_test, y_pred).round(2))
```

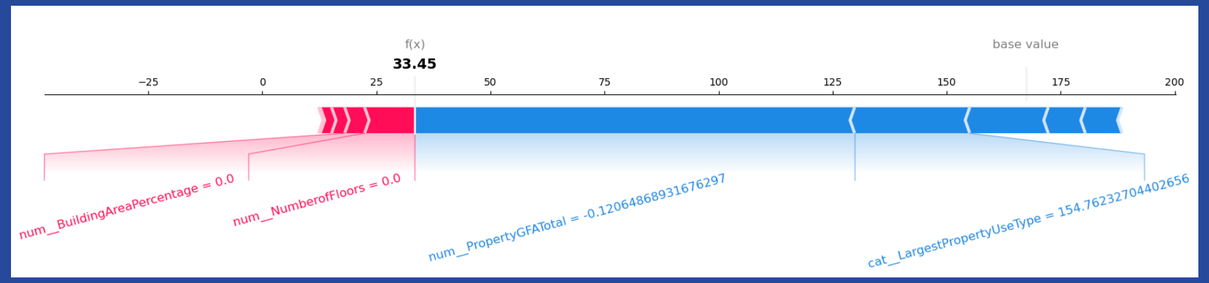
```
R² (Test Set) : 0.6649790735254058
```

Importance des variables (SHAP) emission

Variables les plus importante dans notre dataset



Impact des variables sur un individu



higher ⇄ lower

Conclusion

1. Le modèle arrive en effet à **prédire avec précision la consommation énergétique** des bâtiments non destinés à l'habitation.
2. Les prédictions sur le **modèles des émissions de gaz à effet de serre** sont beaucoup **moins précises**
3. La variable **ENERGYSTARScore augmente la précision dans Les deux modèles**. Cela fait d'elle une variable importante.

Aller plus loin

1. Supprimer les variables qui n'ont pas d'impact sur la précision du modèles
2. Créer et/ou ajouter des variables qui pourrait potentiellement améliorer la précision.
3. Découper et améliorer la fonction : **run_models_with_preprocessors**. afin de gagner en temps lors de son exécution.

Seattle

Merci

Pour votre attention

Source database
data.seattle.gov