PROJET4 ANTICIPEZ LES BESOINS EN CONSOMMATION DE BÂTIMENTS

Apprendre des modèles de Machine Learning

AL SAMMAN Wassim – Data Scientist Apprenti

Mentor : Panayotis PAPOUTSIS

INTRODUCTION ET PROBLÉMATIQUE

- Atteindre l'objectif de la ville neutre de Seattle d'ici 2050.
- Anticiper la consommation d'énergie et les émissions de carbones des bâtiments non destinés à l'habitation.
- Evaluer l'intérêt de la feature Energy Star Score.
- Creuser dans les détails des données pour bien anticiper les deux targets.
- La ville prendra des nouvelles mesures pour atteindre son but.

ANALYSE EXPLORATOIRE
PREMIÈRE PRÉPARATION

- 2 points clés dans les données :
  - ComplianceStatus
  - Outlier

```
df['Outlier'].value_counts()
```

```
Low outlier     23
High outlier     9
Name: Outlier, dtype: int64
```

```
df['ComplianceStatus'].value_counts()
```

```
Compliant                        3211
Error - Correct Default Data      113
Non-Compliant                      37
Missing Data                       15
Name: ComplianceStatus, dtype: int64
```

- Le clé Compliant est suffisant.

```
df=df.loc[df['ComplianceStatus']=='Compliant']
```

```
df['Outlier'].unique() # ne contient pas des valeurs aberrantes
```

```
array([nan], dtype=object)
```

- Choix des targets :
  - GHGEmissionsIntensity ou TotalGHGEmissions ?
  - SiteEUIWN ou SiteEnergyUseWN ?

ANALYSE EXPLORATOIRE
LA FONCTION DF(CHOICE)

- Pourquoi j'ai créé la foncntion df(Choice) ?
- Pourquoi le parking n'est pas important?
- ytarget sans parking
- Problématique des années YearBuilt
- DataFrame par catégorie

```
df('BuildingType') # df par catégo
```

index	SiteEUIWN(kBtu/sf)	SiteEU
campus	2562.299999	
multifamily hr (10+)	5813.499996	
multifamily lr (1-4)	36858.899996	
multifamily mr		

```
dfSurfaceParking['PercentageSurfaceParking'].mean()
```

```
0.042829886453555045
```

	GHGEmissionsIntensityWithoutParking	SiteEUIWN(kBtu/sf)WithoutParking
0	2.830000	84.300003
1	2.444004	83.660139
2	1.739411	77.598389

```
df(4) # df avec tranches des années 3
```

```
Combien d'années par tranche vous souhaitez entre(1-112):3
```

	SiteEUIWN(kBtu/sf)	SiteEUIWN(kBtu/sf)Parking	SiteEUIWN(kBtu/sf)WithoutPa
[1900, 1901, 1902]	4442.700008	74.309127	4368.39
[1903, 1904, 1905]	1337.000002	0.000000	1337.00
[1906,			

ANALYSE EXPLORATOIRE
FEATURE ENGINEERING

- La demande de l'évaluateur
- La fonction feature_engineering(Variable)
- La possibilité de changer le nombre d'année

```
FETop5=feature_engineering('Top5')
FETop5
```

	LargestPropertyUseType	LargestPropertyUseTypeGFA	GHGEmissionsIntensityWithoutParking	SiteEUIWN(kBtu/sf)WithoutParking	FirstSurface %	S
0	Office	59617278.0	432.372809	28000.039236	76.05	
1	Other	8024345.0	185.760810	8728.151283	10.24	
2	Parking	4439882.0	20.183324	2264.526351	5.66	
3	Restaurant	319611.0	81.179261	2312.581802	0.41	
4	Retail Store	5996141.0	123.930619	6304.263481	7.65	

```
feature_engineering('Neighborhood')
```

index	ballard	central	delridge	del neighborh
0	1.0	0.0	0.0	
1	0.0	1.0	0.0	

```
feature_engineering('CategoryYears')
Combien d'années par tranche vous sou
```

MaxYear	1909	1919	1929	1962	1972	198
0	1.0	0.0	0.0	0.0	0.0	0.
1	0.0	1.0	0.0	0.0	0.0	0.

```
feature_engineering('Binary')
```

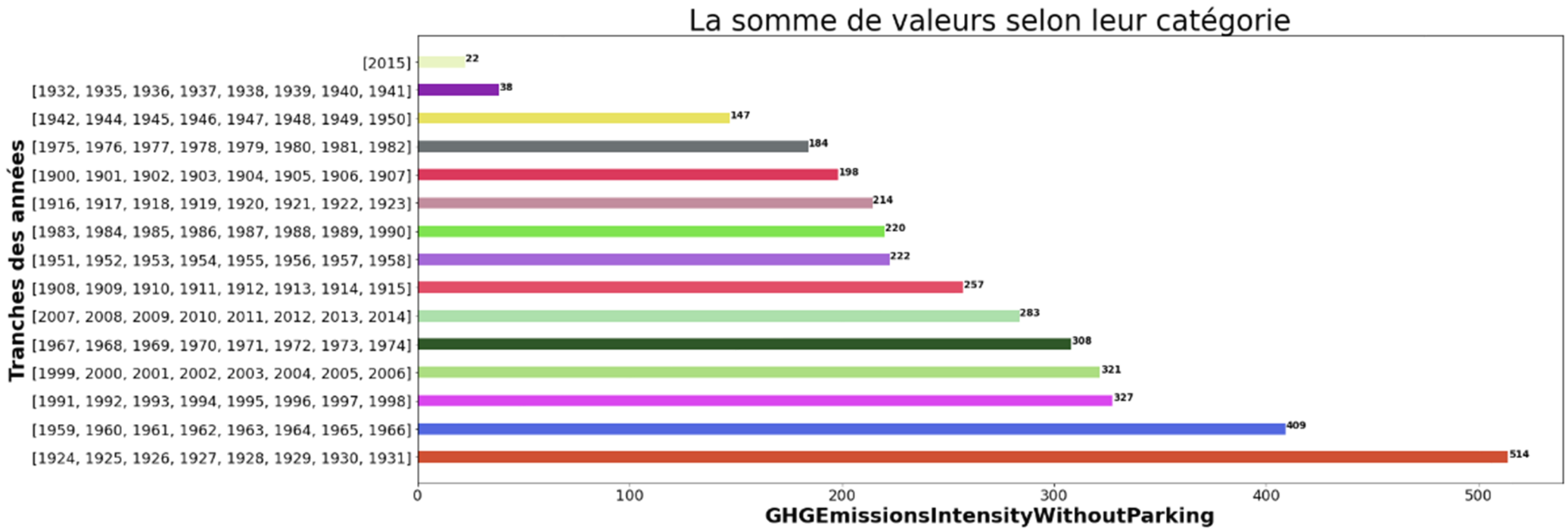
	BinaryNaturalGas(kBtu)	BinarySteamUse(kBtu)
0	1	1
1	1	0

ANALYSE EXPLORATOIRE

PRÉSENTER YTARGET

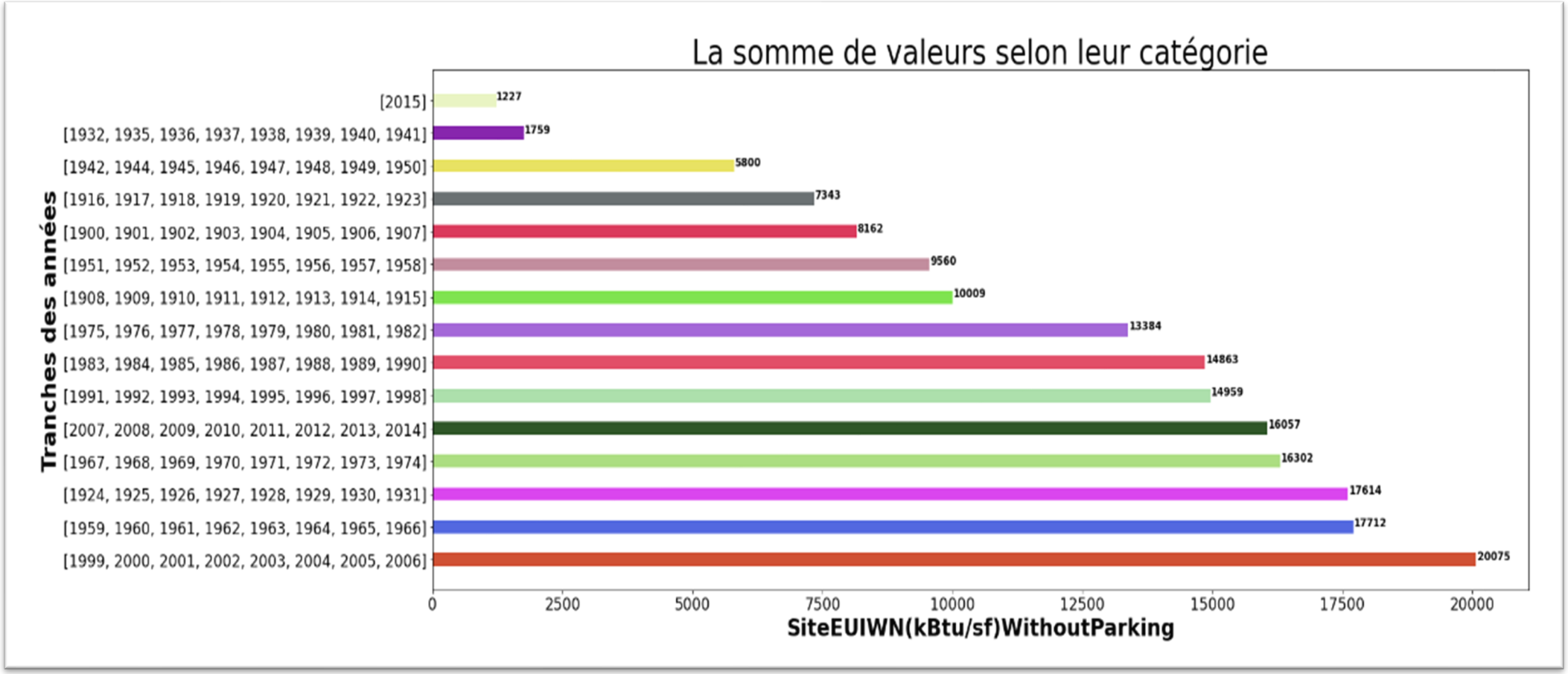
```
barplot(4,10);
```

```
Combien d'années par tranche vous souhaitez entre(1-112):8
```



ANALYSE EXPLORATOIRE

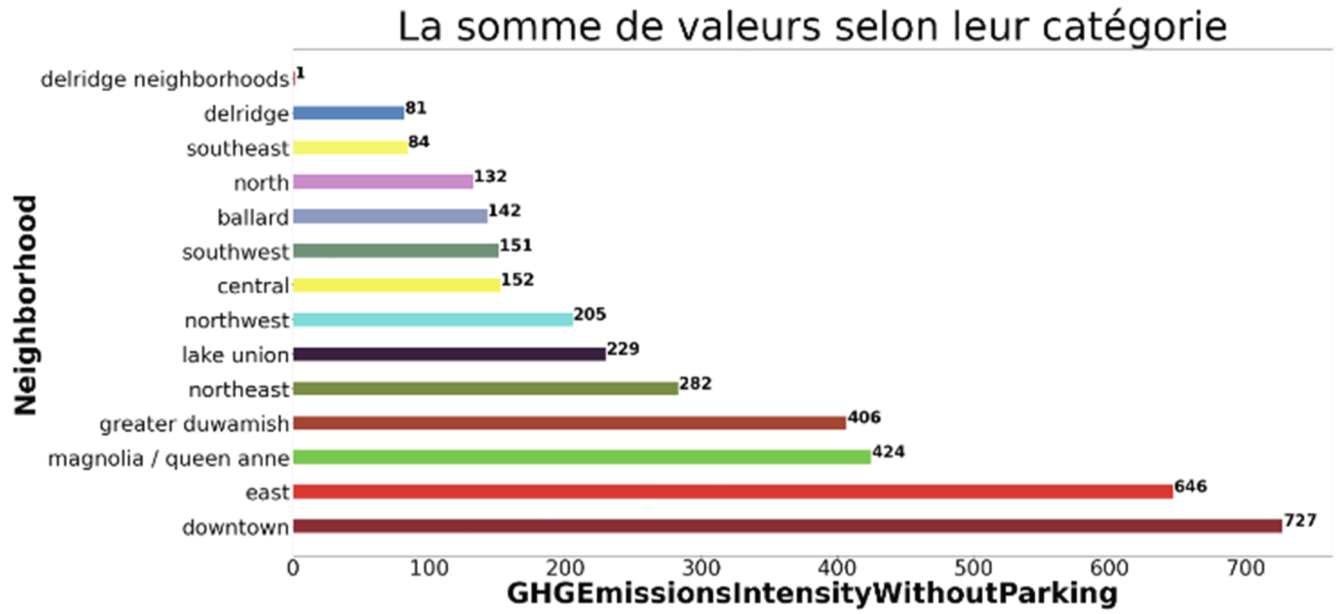
PRÉSENTER YTARGET



ANALYSE EXPLORATOIRE

EXEMPLE DE VARIABLES X ET Y

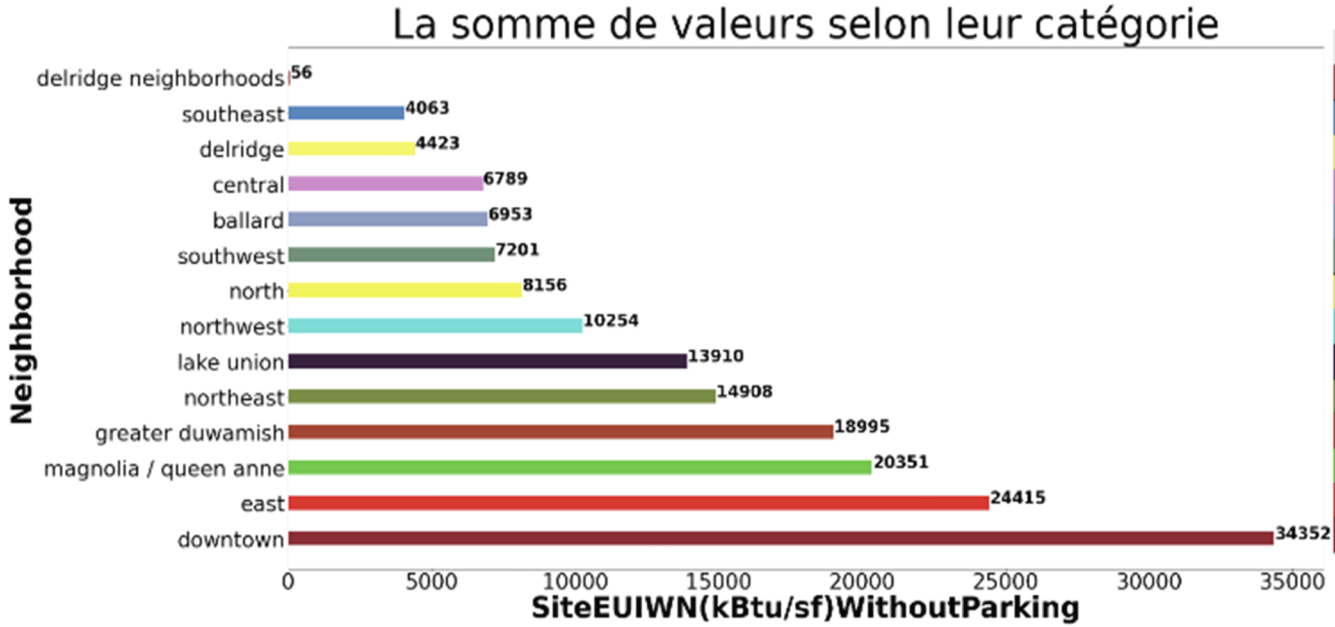
```
barplotwithtable('Neighborhood',20);
```



NumberofFloors	NumberofBuildings	PropertyGFABuilding(s)
1.0	1.0	18261.0
202.0	113.0	5838827.0
241.0	82.0	5270388.0
604.0	186.0	12242300.0
449.0	136.0	8357905.0
594.0	145.0	6873076.0
436.0	136.0	7420589.0
692.0	260.0	13284227.0
1069.0	252.0	22918810.0
1022.0	411.0	26470785.0
770.0	394.0	26294079.0
1670.0	456.0	25179124.0
2284.0	449.0	33241963.0
5394.0	543.0	87331301.0

ANALYSE EXPLORATOIRE

EXEMPLE DE VARIABLES X ET Y



NumberofFloors	NumberofBuildings	PropertyGFABuilding(s)
1.0	1.0	18261.0
241.0	82.0	5270388.0
202.0	113.0	5838827.0
436.0	136.0	7420589.0
449.0	136.0	8357905.0
594.0	145.0	6873076.0
604.0	186.0	12242300.0
692.0	260.0	13284227.0
1069.0	252.0	22918810.0
1022.0	411.0	26470785.0
770.0	394.0	26294079.0
1670.0	456.0	25179124.0
2284.0	449.0	33241963.0
5394.0	543.0	87331301.0

RÉGRESSION LINÉAIRE

- La fonction *linear_regression()*

```
def linear_regression(Method,yNumericalColumnsToFill=ytarget,XNumericalColumn=AllNumericalX,
                      Alpha=1,EndRangeAlphaParametre=2):
```

- DataFrame rempli avec le modèle choisi

```
linear_regression(1)[ytarget].isna().sum()
```

```
GHGEmissionsIntensityWithoutParking    0
SiteEUIWN(kBtu/sf)WithoutParking       0
ENERGYSTARScore                        0
dtype: int64
```

- Avoir les valeurs prévues de tous les modèles pour les comparer

```
linear_regression(6)[1] # Avoir un seul DataFrame
```

	LinearRegression()	Lasso()	Ridge()	ElasticNet()	DummyRegressor()	SiteEUIWN(kBtu/sf)WithoutParking
513	46.954521	54.378522	56.019302	54.310399	54.378522	NaN

```
linear_regression(6,XNumericalColumn=['YearBuilt'])[1]
```

	LinearRegression()	Lasso()	Ridge()	ElasticNet()	DummyRegressor()	SiteEUIWN(kBtu/sf)WithoutParking
513	53.436952	54.378522	53.442157	54.324801	54.378522	NaN

- Utiliser train_test_split ?

RÉGRESSION LINÉAIRE

- Trouver le meilleur alpha
- Calculer les erreurs:
  - Mean Absolute Error (MAE)
  - Root Mean Square Error (RMSE)
  - R-squared (R2)

```
linear_regression(7) # Trouver meilleur alpha
```

	GHGEmissionsIntensityWithoutParking_BestAlpha	Si
Lasso()	1.0	
Ridge()	1.9	
ElasticNet()	1.0	

```
linear_regression(8) # évaluer la performance
```

	GHGEmissionsIntensityWithoutParking_MAE	GHGEmissionsIntensityWithoutParking_RMSE	GHGEmissionsIntensityWithoutParking_R2	S
LinearRegression()	0.977434	1.834929	0.035356	
Lasso(alpha=1)	1.030038	1.868266	-0.000013	
Ridge(alpha=1)	0.980916	1.839525	0.030517	
ElasticNet(alpha=1)	1.030038	1.868266	-0.000013	
DummyRegressor()	1.030038	1.868266	-0.000013	

RÉGRESSION LINÉAIRE

- Trouver la meilleure feature pour un modèle

```
def choose_best_feature(y_target,Alpha=1,HowManyBestFeature=1):
```

```
choose_best_feature(y_target) # y_target='GHGEmissionsIntensityWithoutParking'
```

	LargestPropertyUseTypeGFA	SecondLargestPropertyUseTypeGFA	ThirdLargestPropertyUseTypeGFA	With_y
LinearRegression()	1	2	3	GHGEmissionsIntensityWithoutParking
Lasso(alpha=1)	3	2	1	GHGEmissionsIntensityWithoutParking
Ridge(alpha=1)	1	2	3	GHGEmissionsIntensityWithoutParking
ElasticNet(alpha=1)	3	2	1	GHGEmissionsIntensityWithoutParking

RÉGRESSION NON-LINÉAIRE

- La fonction *non_linear_regression()*

```
def non_linear_regression(yNumericalColumnsToFill=ytarget,Regression='Trees',XNumericalColumn=AllNumericalX):
```

- Les modèles de la variable Regression ‘Trees’ et ‘OneTree’
- Comparer les résultats

```
non_linear_regression(Regression='Compare')[1]
```

	Trees	OneTree	SiteEUIWN(kBtu/sf)WithoutParking
513	67.932419	58.487216	NaN

```
non_linear_regression(Regression='Compare',XNumericalColumn=['YearBuilt'])[1]
```

	Trees	OneTree	SiteEUIWN(kBtu/sf)WithoutParking
513	48.881428	50.80813	NaN

RÉGRESSION NON-LINÉAIRE

- Trouver les meilleures paramètres

```
non_linear_regression(Regression='BestForTrees') # Les meilleurs paramètres pour '
```

	learning_rate	max_depth	min_split_loss	sampling_method
GHGEmissionsIntensityWithoutParking	0.1	5	0	uniform
SiteEUIWN(kBtu/sf)WithoutParking	0.1	5	0	uniform
ENERGYSTARScore	0.1	5	1	uniform

```
non_linear_regression(Regression='BestForOneTree') # Les meilleurs paramètres pour
```

	alpha	learning_rate	loss	max_depth
GHGEmissionsIntensityWithoutParking	0.7	0.1	squared_error	2
SiteEUIWN(kBtu/sf)WithoutParking	0.7	0.1	squared_error	2
ENERGYSTARScore	0.7	0.1	squared_error	2

- Calculer le score KFold et les erreurs (MAE,RMSE,R2)

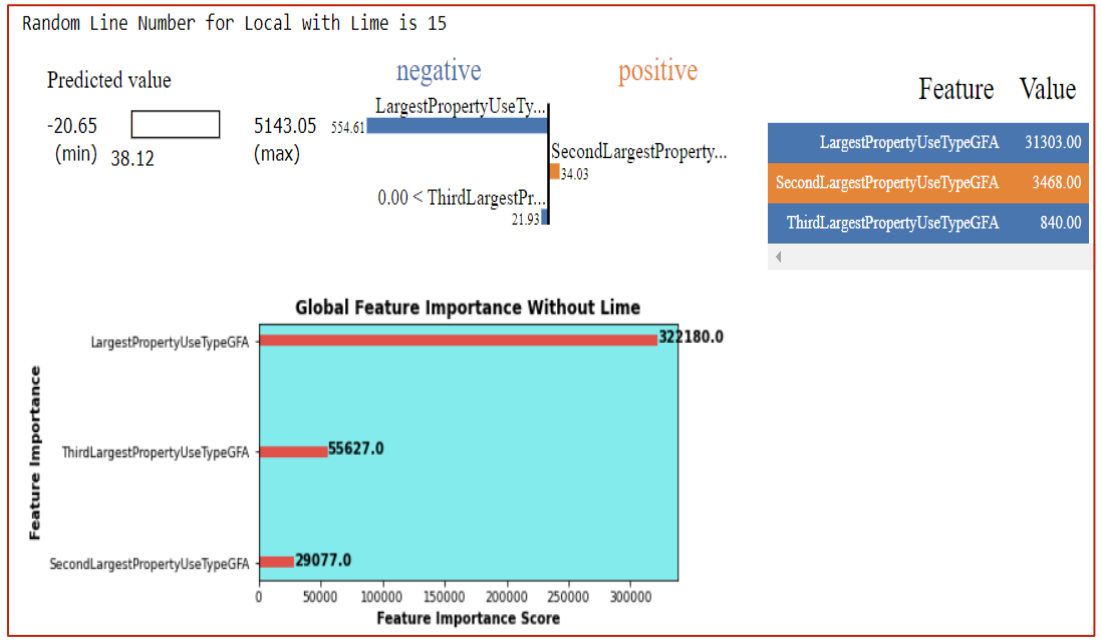
```
error=non_linear_regression(Regression='errors')
error
```

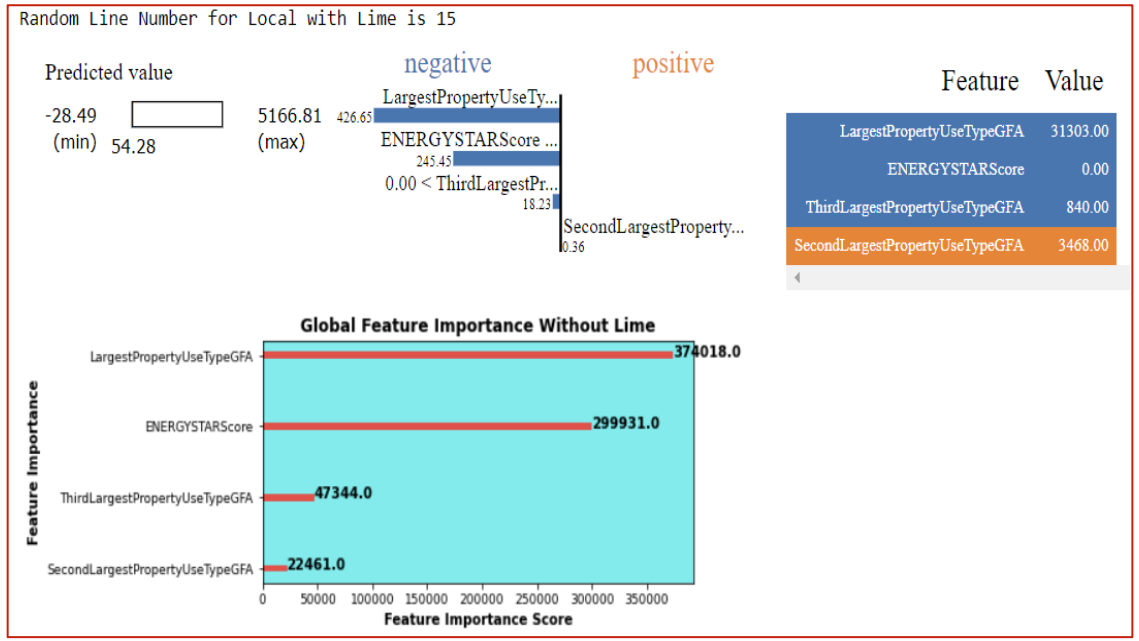
	GHGEmissionsIntensityWithoutParking_AccuracyMean(%)	GHGEmissionsIntensityWithoutParking_MAE	GHGEmissionsIntensityWithoutParking_RMSE
Trees	13.845916	0.913680	1.906967
OneTree	11.509591	0.912611	1.836059

RÉGRESSION NON-LINÉAIRE

- Feature importance ‘Trees’ local et global pour le deuxième target avec la fonction:

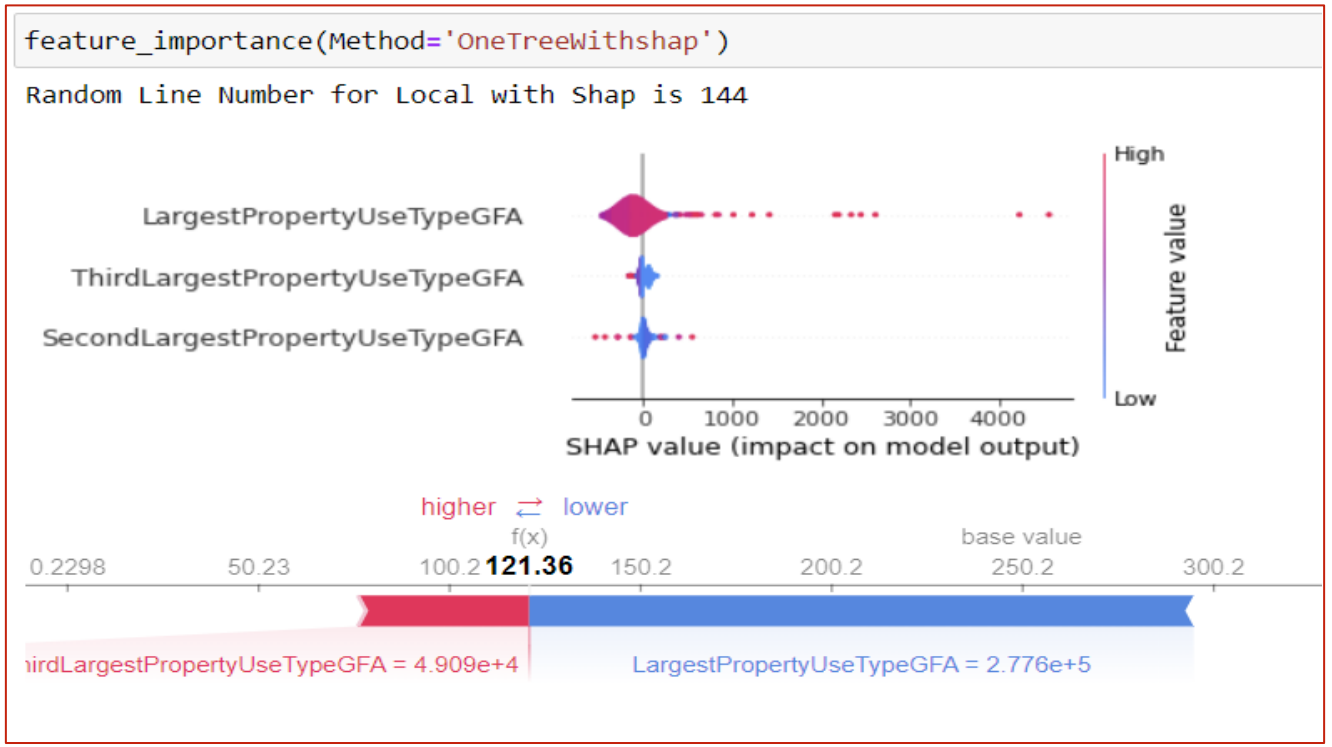
```
def feature_importance(Method,y_target=ytarget,ChooseSpecificLineNumber=None):
```





RÉGRESSION NON-LINÉAIRE

- Feature importance ‘OneTree’ local et global pour le deuxième target



CONCLUSION

- Ce que j’ai appris
- La partie intéressante pour moi
- Mon sentiment à la moitié de la formation