M2 - Probabilités et statistiques des nouvelles données

Projet : Kaggle " House Prices"

Auteur :
Confiac Hendrick

Février 2022

Préface

Ce Projet a pour objectif de développer un modèle de machine learning afin de prédire au mieux le prix de logements. Sur Kaggle, 3 datasets nous sont mis a disposition. Notre modèle, que l'on cherchera a optimiser, s'entrainera et sera évalué sur le dataset nommé 'train' (*traindf* sur notre notebook). afin de soumettre une prédiction des prix de logements à partir des données du dataset 'test' (testdf dans notre notebook). Pour mener a bien ce projet, nous avons procédé par étape dans le but d'optimiser la performance finale de notre modèle de machine learning.

NB : Pour des soucis de respect de pages, la liste de plots qui vont paraîtres dans ce projet est non exhaustive et se trouve dans notre Notebook. Ici nous y mettrons les plots les plus "pertinents".

Table des matières

Chapitre 1

Exploration des données

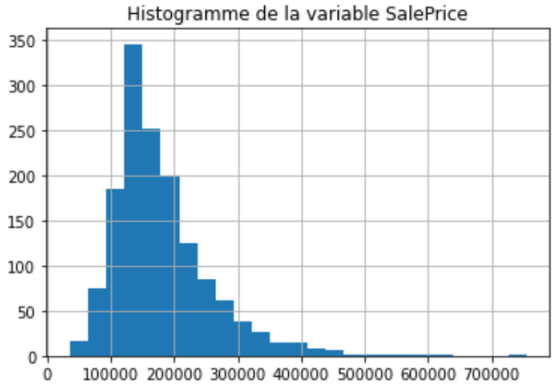
Le but est de s'approprier les données du dataset sous toutes ses coutures et connaître les caractéristiques de ce dernier.

1.1 Analyse en surface du Dataset :

Description des données :

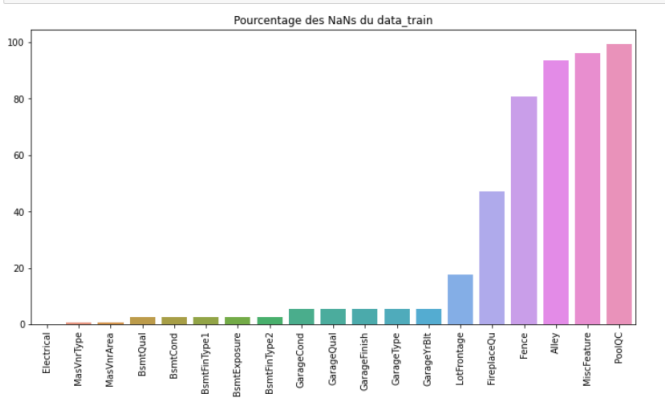
Vous trouver ci-joints l'ensemble détaillé des features qui composent notre dataset. `https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data`

— SalePrice - le prix de vente de la propriété en dollars. Il s'agit de notre target.



Nous travaillons donc avec $data_train$, dataset composé de 1460 lignes et 80 colonnes. On y retourve 37 variables quantitatives, dont 12 continues et 25 discrètes et 43 variables qualitatives. Parmi ces variables, certaines possèdent des valeurs manquantes.

Electrical	0.000685
MasVnrType	0.005479
MasVnrArea	0.005479
BsmtFinType1	0.025342
BsmtCond	0.025342
BsmtQual	0.025342
BsmtExposure	0.026027
BsmtFinType2	0.026027
GarageYrBlt	0.055479
GarageQual	0.055479
GarageFinish	0.055479
GarageCond	0.055479
GarageType	0.055479
LotFrontage	0.177397
FireplaceQu	0.472603
Fence	0.807534
Alley	0.937671
MiscFeature	0.963014
PoolQC	0.995205



Certaines 'features' ont la même proportion de variables manquantes. Parmi celles-ci on distingue :

— GarageQual GarageFinish GarageYrBlt GarageType GarageCond $(5,5479\%)$ et BsmtQual BsmtCond BsmtFinType1 $(2,5342\%)$ qui selon kaggle font référence au fait que le logement étudié ne possède pas la caractéristiqe en question (pas de garage/ pas de sous-sol). Par exemple, $99,5205\%$ des logements dans notre étude n'ont pas de piscine (PoolQC).

— MasVnrType MasVnrArea (0.5479%) représentant un manque de données chez ces variables pour le logement étudié.

1.2 Analyse en profondeur :

Corrélation entres variables quantitatives

Déterminons les variables dont la corrélation avec SalePrice (notre target) est supérieur à $0,5$:

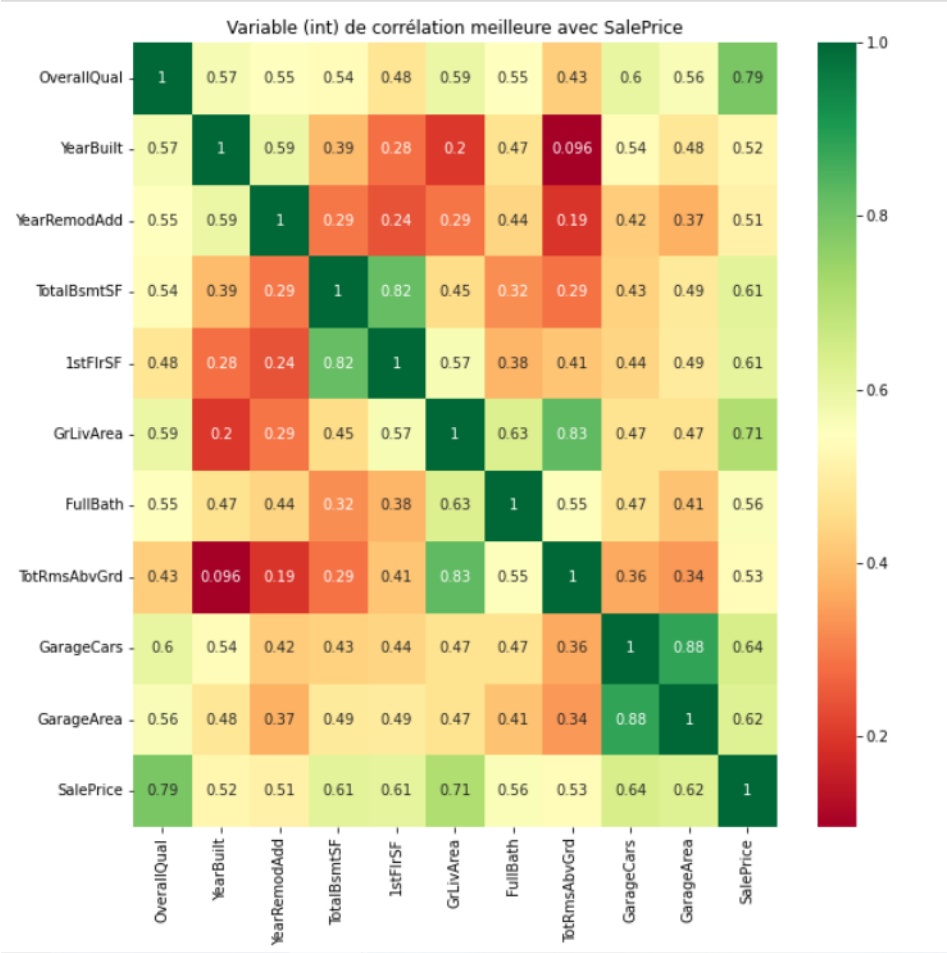


FIGURE 1.1 – Variables de corrélation meilleure avec 'SalePrice'

Notons que Overquall est la variable la plus corrélée avec SalePrice (0.79). Par ailleurs certaines variables sont de corrélation très élevées entre elles. Ceci pouvant se voir par la couleur verte de la case qui relie ces deux variables à l'instar de GarageCars et GarageArea (0.88) pour ne citer qu'eux. Visualisons graphiquement ces résultats :

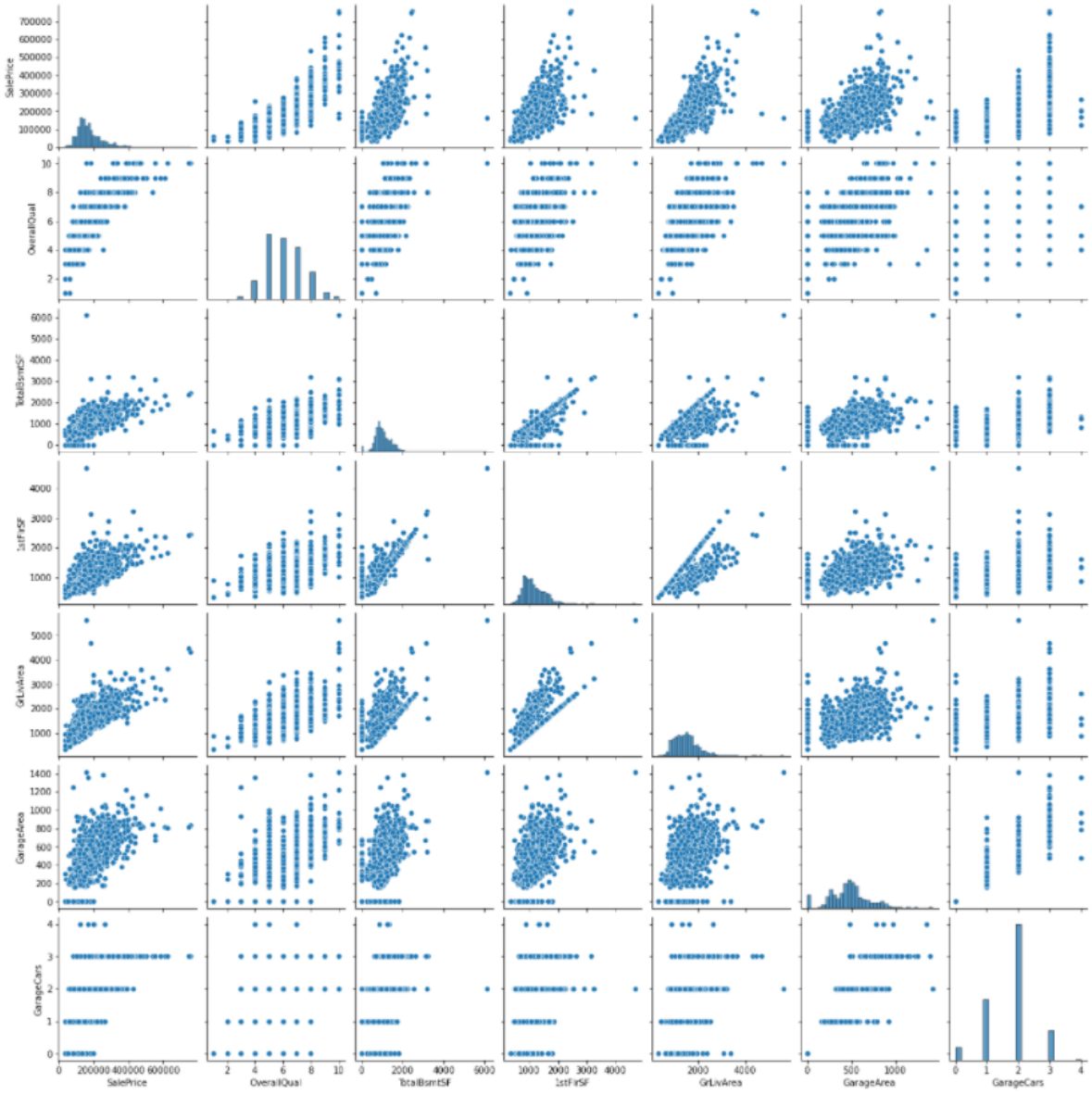
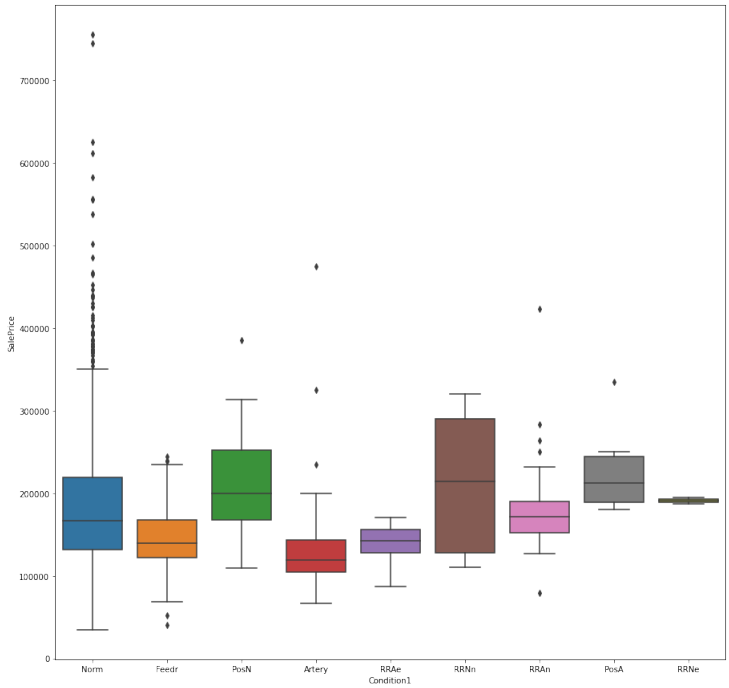
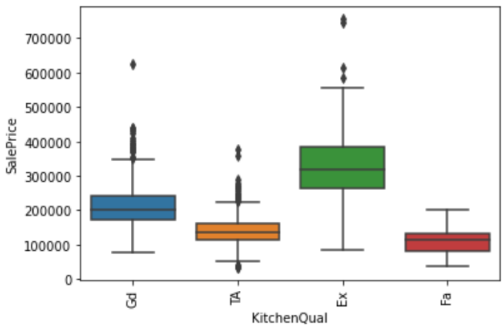


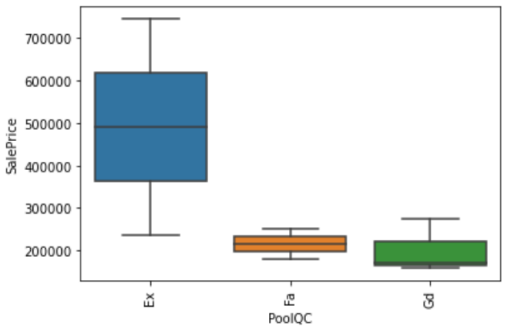
FIGURE 1.2 – Dépendances linéaires des variables de corrélation meilleure avec 'SalePrice'

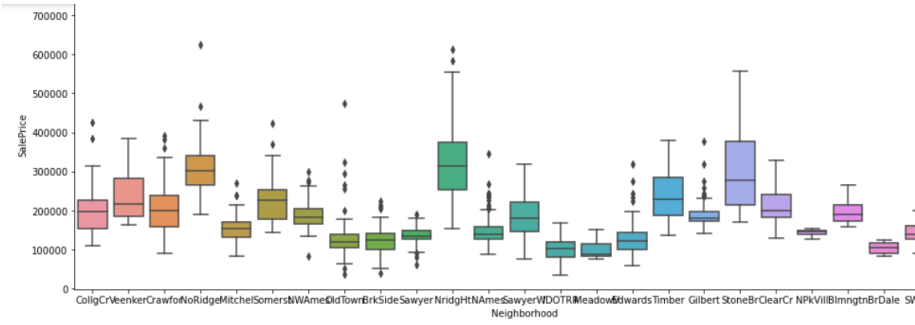
Relations des variables qualitatives/SalePrice

Nous avons relevé quelques variables ayant vraisemblablement une influence sur le prix de vente des logements. Parmi elles nous retrouvons 'PoolQC', 'KitchenQual','SaleType','Neighborhood'... :









À travers ces graphiques nous pouvons lire que les maisons les plus chères sont en présence d'une piscine de qualité qualifié de 'Excellente' ou encore que le prix du logement est d'autant plus élevé que la qualité de la cuisine est bonne. Neighboorhood et Condition1 semblent avoir une grande influence sur le prix de vente.

Chapitre 2

Pre-Processing

Après avoir pris un peu plus connaissance sur les caractéristiques de notre dataset (variables importantes, variables corrélées, nombre de valeurs manquantes...), nous allons passer à la partie Pre-Processing, étape permettant à notre Dataset d'être 'éligible'à l'application dans un algorithme de Machine Learning

2.1 Elimination des données manquantes

La première étape est de traiter les valeurs manquantes .

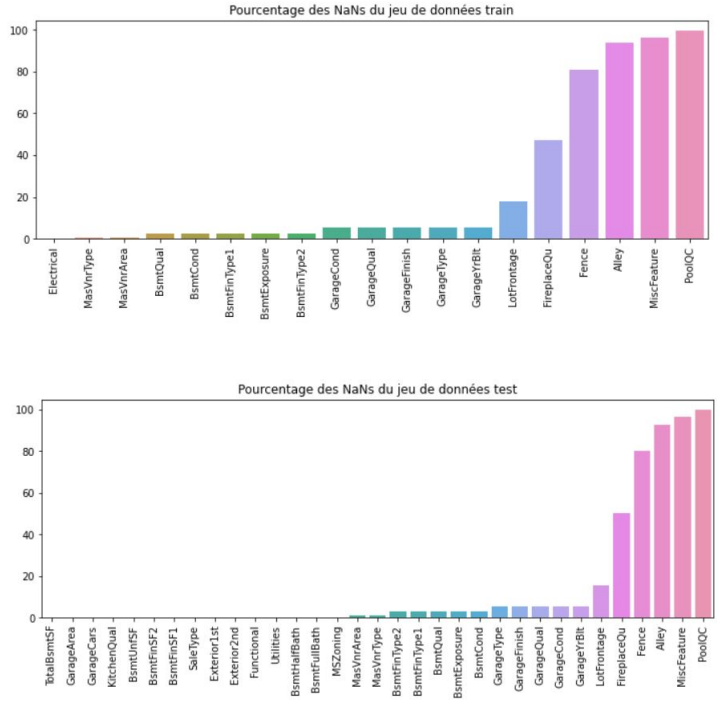
	LotFrontage	Alley	MasVnrType	MasVnrArea	BsmtQual	BsmtCond	BsmtExposure	BsmtFinType1	BsmtFinType2	Electrical	FireplaceQu	GarageType
0	65.0	NaN	BrkFace	196.0	Gd	TA	No	GLQ	Unf	SBrkr	NaN	Attchd
1	80.0	NaN	None	0.0	Gd	TA	Gd	ALQ	Unf	SBrkr	TA	Attchd
2	68.0	NaN	BrkFace	162.0	Gd	TA	Mn	GLQ	Unf	SBrkr	TA	Attchd
3	60.0	NaN	None	0.0	TA	Gd	No	ALQ	Unf	SBrkr	Gd	Detchd
4	84.0	NaN	BrkFace	350.0	Gd	TA	Av	GLQ	Unf	SBrkr	TA	Attchd
...	...	...	...	...	...	...	...	...	...	...	...	...
1455	62.0	NaN	None	0.0	Gd	TA	No	Unf	Unf	SBrkr	TA	Attchd
1456	85.0	NaN	Stone	119.0	Gd	TA	No	ALQ	Rec	SBrkr	TA	Attchd
1457	66.0	NaN	None	0.0	TA	Gd	No	GLQ	Unf	SBrkr	Gd	Attchd
1458	68.0	NaN	None	0.0	TA	TA	Mn	GLQ	Rec	FuseA	NaN	Attchd
1459	75.0	NaN	None	0.0	TA	TA	No	BLQ	LwQ	SBrkr	NaN	Attchd

1460 rows x 19 columns

GarageYrBlt	GarageFinish	GarageQual	GarageCond	PoolQC	Fence	MiscFeature
2003.0	RFn	TA	TA	NaN	NaN	NaN
1976.0	RFn	TA	TA	NaN	NaN	NaN
2001.0	RFn	TA	TA	NaN	NaN	NaN
1998.0	Unf	TA	TA	NaN	NaN	NaN
2000.0	RFn	TA	TA	NaN	NaN	NaN
...	...	...	...	...	...	...
1999.0	RFn	TA	TA	NaN	NaN	NaN
1978.0	Unf	TA	TA	NaN	MnPrv	NaN
1941.0	RFn	TA	TA	NaN	GdPrv	Shed
1950.0	Unf	TA	TA	NaN	NaN	NaN
1965.0	Fin	TA	TA	NaN	NaN	NaN

FIGURE 2.1 – DataFrame Colonnes aux données manquantes de traindf

Visualisation de la proportion de données manquantes de traindf et testdf



Imputation

Nous allons procéder par étape. Tout d'abord occupons-nous des NaNs des variables, qui, selon kaggle, corresponderaient au fait que le logement étudié ne possède pas la caractéristique en question.

Imputation variables aux caractéristiques absentes

Nous allons donc préciser dans nos données que ces caractéristiques sont manquantes.Nous allons faire cette opération sur nos deux dataset en les concatenant et en appellant ce nouveau dataset 'data' ; Par ailleurs puisque la caractéristique est manquante dans le logement associé, les données manquantes des variables quantifiants ces caractéristiques seront remplacées par 0 :

```
# Pas d'Allée
data['Alley'].fillna("Nothing", inplace=True)

#Pas de sous-sol
data['BsmtCond'].fillna("NoB", inplace=True)
data['BsmtExposure'].fillna("NoB", inplace=True)
data['BsmtFinType1'].fillna("NoB", inplace=True)
data['BsmtFinType2'].fillna("NoB", inplace=True)
data['BsmtQual'].fillna("NoB", inplace=True)

#Pas de cheminé
data['FireplaceQu'].fillna("None", inplace=True)

#Pas de Garage
data['GarageType'].fillna("No", inplace=True)
data['GarageFinish'].fillna("No", inplace=True)
data['GarageQual'].fillna("No", inplace=True)
data['GarageCond'].fillna("No", inplace=True)

#Pas de Piscine
data['PoolQC'].fillna("NoP", inplace=True)

#pas de clôture
data['Fence'].fillna("NoFence", inplace=True)

#pas "d'avantages" supplémentaires
data['MiscFeature'].fillna("NoM", inplace=True)

data['MasVnrType'].fillna("NoMasVnr",inplace=True)
```

```
for i in data[['GarageYrBlt', 'GarageArea','GarageCars','MasVnrArea', 'GarageYrBlt','BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF',
       'TotalBsmtSF', 'BsmtFullBath', 'BsmtHalfBath']].columns:
    data[i].fillna(0, inplace=True)
```

Imputation variables par leurs catégories

Cette intuition est motivée par l'analyse et la mise en relation chez certaines variables.
Par exemple : Tout logement est équipé d'une installation électrique. Ceci implique que la donnée manquante de la catégorie Electrical serait dûe a une absence d'informations.

	YearBuilt	Electrical	Utilities
1379	2006	NaN	AllPub

NB : La catégorie AllPub de la variable Utilities correspond au fait que tous les services publiques (eau ,électricité, gaz..) sont présents dans le logement associé.

Pour les variables explicatives de type 'entier' nous remplacerons les valeurs nulles par la moyenne sur cette variables.

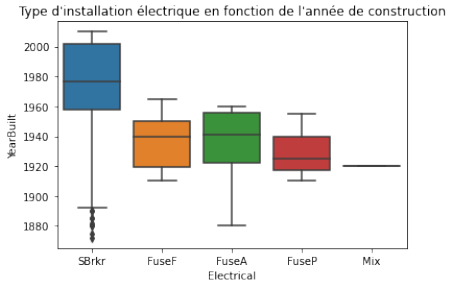


FIGURE 2.2 – Relation Electrical/YearBuilt

De plus, il y aurait une influence de la variable 'YearBuilt' sur la variable 'Electrical' selon le boxplot. Par ailleurs l'année de construction du logement associé est 2006 d'où notre choix de remplacer la valeur manquante de 'Electrical' par : 'SBrkr', seule catégorie présente dans les années 2000.

d

L'imputation de LotFrontage se fera en prenant la moyenne des données de celle-ci.

```
# Pour LotFrontage
train_df['LotFrontage'].fillna(train_df['LotFrontage'].mean(), inplace=True)
test_df['LotFrontage'].fillna(test_df['LotFrontage'].mean(), inplace=True)
```

L'imputation de LotFrontage se fera en prenant la moyenne des données de celle-ci.

Nous allons maintenant automatiser le remplacement des données manquantes grâce à la méthode de KNNImputer implentée dans notre FONCTION IMPUT :
Nous allons créer une fonction qui va dans un premier temps, labéliser les variables de type objet dont on souhaite éliminer les NaNs, puis fera intervenir KNNImputer afin de remplacer les données manquantes en prenant (ici n neighbors=1) le voisin le plus proche. Puis nous allons faire une boucle dans laquelle nous ferrons appelle à la fonction pour chaque variables restantes ayant des NaNs.

```
train_df.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...	ScreenPorch	PoolArea	PoolQC	Fence	
0	60	RL	65.0	8450	Pave	Nothing	Reg	Lvl	AllPub	Inside	...	0	0	NoP	NoFence	
1	20	RL	80.0	9600	Pave	Nothing	Reg	Lvl	AllPub	FR2	...	0	0	NoP	NoFence	
2	60	RL	68.0	11250	Pave	Nothing	IR1	Lvl	AllPub	Inside	...	0	0	NoP	NoFence	
3	70	RL	60.0	9550	Pave	Nothing	IR1	Lvl	AllPub	Corner	...	0	0	NoP	NoFence	
4	60	RL	84.0	14260	Pave	Nothing	IR1	Lvl	AllPub	FR2	...	0	0	NoP	NoFence	

5 rows × 79 columns

```
test_df.head()
```

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...	ScreenPorch	PoolArea	PoolQC	Fence	
0	20	2.0	80.0	11622	Pave	Nothing	Reg	Lvl	0.0	Inside	...	120	0	NoP	MnPrv	
1	20	3.0	81.0	14267	Pave	Nothing	IR1	Lvl	0.0	Corner	...	0	0	NoP	NoFence	
2	60	3.0	74.0	13830	Pave	Nothing	IR1	Lvl	0.0	Inside	...	0	0	NoP	MnPrv	
3	60	3.0	78.0	9978	Pave	Nothing	IR1	Lvl	0.0	Inside	...	0	0	NoP	NoFence	
4	120	3.0	43.0	5005	Pave	Nothing	IR1	HLS	0.0	Inside	...	144	0	NoP	NoFence	

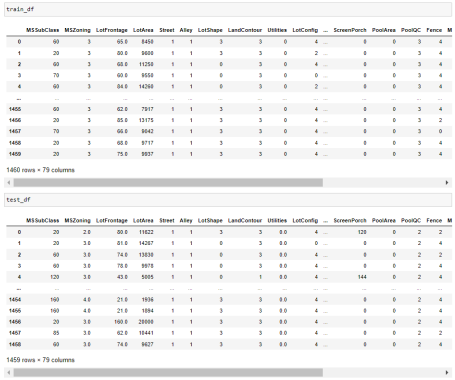
5 rows × 79 columns

```
# Verifions que les données manquantes ont été remplacés
train_df.isnull().sum().sum()
```

```
0
```

2.2 Label Encoding et standardisation

Nous allons procéder à l'encodage des variables catégorielles puis à la standardisation des variables quantitatives. Les transformers Label.Encoder() et StandardScaler() seront nos outils dans cette procédure.



train_df

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...	ScreenPorch	PoolArea	PoolQC	Fence	M
0	60	3	65.0	8450	1	1	3	3	0	4	...	0	0	3	4	
1	20	3	80.0	9600	1	1	3	3	0	2	...	0	0	3	4	
2	60	3	68.0	11250	1	1	0	3	0	4	...	0	0	3	4	
3	70	3	60.0	9550	1	1	0	3	0	0	...	0	0	3	4	
4	60	3	84.0	14260	1	1	0	3	0	2	...	0	0	3	4	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1455	60	3	62.0	7917	1	1	3	3	0	4	...	0	0	3	4	
1456	20	3	85.0	13175	1	1	3	3	0	4	...	0	0	3	2	
1457	70	3	66.0	9042	1	1	3	3	0	4	...	0	0	3	0	
1458	20	3	68.0	9717	1	1	3	3	0	4	...	0	0	3	4	
1459	20	3	75.0	9937	1	1	3	3	0	4	...	0	0	3	4	

1460 rows × 79 columns

test_df

	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	LotConfig	...	ScreenPorch	PoolArea	PoolQC	Fence	M
0	20	2.0	80.0	11622	1	1	3	3	0.0	4	...	120	0	2	2	
1	20	3.0	81.0	14267	1	1	0	3	0.0	0	...	0	0	2	4	
2	60	3.0	74.0	13830	1	1	0	3	0.0	4	...	0	0	2	2	
3	60	3.0	78.0	9978	1	1	0	3	0.0	4	...	0	0	2	4	
4	120	3.0	43.0	5005	1	1	0	1	0.0	4	...	144	0	2	4	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1454	160	4.0	21.0	1936	1	1	3	3	0.0	4	...	0	0	2	4	
1455	160	4.0	21.0	1894	1	1	3	3	0.0	4	...	0	0	2	4	
1456	20	3.0	160.0	20000	1	1	3	3	0.0	4	...	0	0	2	4	
1457	85	3.0	62.0	10441	1	1	3	3	0.0	4	...	0	0	2	2	
1458	60	3.0	74.0	9627	1	1	3	3	0.0	4	...	0	0	2	4	

1459 rows × 79 columns

FIGURE 2.3 – traindf et testdf après encodage

	MSSubClass	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	1stFlrSF	2ndFlrSF	LowQualFinSF	GrLivArea	...	MasVnrArea	BsmtFinSF1
0	0.073375	-0.207142	0.651479	-0.517200	1.050994	0.878668	-0.793434	1.161852	-0.120242	0.370333	...	0.514104	0.575425
1	-0.872563	-0.091886	-0.071836	2.179628	0.156734	-0.429577	0.257140	-0.795163	-0.120242	-0.482512	...	-0.570750	1.171992
2	0.073375	0.073480	0.651479	-0.517200	0.984752	0.830215	-0.627826	1.189351	-0.120242	0.515013	...	0.325915	0.092907
3	0.309859	-0.096897	0.651479	-0.517200	-1.863632	-0.720298	-0.521734	0.937276	-0.120242	0.383659	...	-0.570750	-0.499274
4	0.073375	0.375148	1.374795	-0.517200	0.951632	0.733308	-0.045611	1.617877	-0.120242	1.299326	...	1.366489	0.463568

5 rows × 36 columns

FIGURE 2.4 – traindf après normalisation

Notre tranidf est enfin prêt à l'usage et notre testdf est prêt à être appliquer à notre modèle de machine learning

2.3 Machine Learning

Nous avons sélectionner 3 prédicteurs de régression :

- RandomForestRegressor()
- Lasso()
- DecisionTree ()

Grâce à la méthode GridSearchCv nous allons pouvoir évaluer la performance du modèle en évaluant ce dernier avec différents hyperparamètres dont on retiendra ceux qui nous donnent le meilleur score. Nous avons défini une fonction qui retourne les meilleurs hyperparamètres parmis ceux figurant dans un dictionnaire qu'on aura défini.

```
def best_hyper_para (model,param):
    grid = GridSearchCV(model,param, cv=4)
    grid.fit(x_train,y_train)
    model_fin = grid.best_estimator_
    return(model_fin, grid.best_score_,grid.best_params_)
```

Nous allons l'appliquer à chaque prédicteurs

— RandomForestRegressor()

```
RandomForestRegressor(max_features='sqrt', n_estimators=700, n_jobs=-1,
                      random_state=8)
```

— Lasso()

```
Lasso(alpha=4.9)
```

— DecisionTree ()

```
DecisionTreeRegressor(criterion='absolute_error', random_state=7)
```

Nous obtenons le résulat final suivant :

	model_name	score_test	MSE
0	model_randomforest	0.924245	5.700761e+08
1	model_randomforest_scaled	0.929884	5.276459e+08
2	model_decisiontree	0.741550	1.944913e+09
3	model_decisiontree_scaled_	0.808895	1.438125e+09
4	model_lasso	0.882149	8.868670e+08
5	model_lasso_scaled	0.873415	9.525929e+08

Conclusion

On voit clairement que le modele "Random Forest" est plus éfficace que les autres , en effet c'est un algorithme particulièrement performant pour les problématiques de prédiction.

C'est un modèle d'apprentissage, dont l'efficacité dépend fortement de la qualité de l'échantillon de données de départ.

Grâce aux algorithmes de Machine learning , Nous avons pu développer des modèles pour prédire les prix des logements .

Bibliographie

[1] Professeur : Romuald ELIE , Cours de machine learning , *Université Gustave Eiffel*(2021)