Evrard Thomas
Pelletier Fabien

Rapport de projet M1 SID

-

House Prices - Advanced Regression Techniques

Introduction

Dans le cadre de ce projet, nous allons étudier un dataset présent sur la plateforme Kaggle : https://www.kaggle.com/c/house-prices-advanced-regression-techniques/overview

L'objectif de ce projet est d'appliquer les connaissances acquises au cours de notre formation afin de prédire le prix de maison en fonction de leurs caractéristiques.

Ce rapport accompagne une vidéo présentant également notre projet.

Prétraitement

NaN

Nous regroupons dans cette partie l'ensemble des techniques et transformations appliquées aux données.

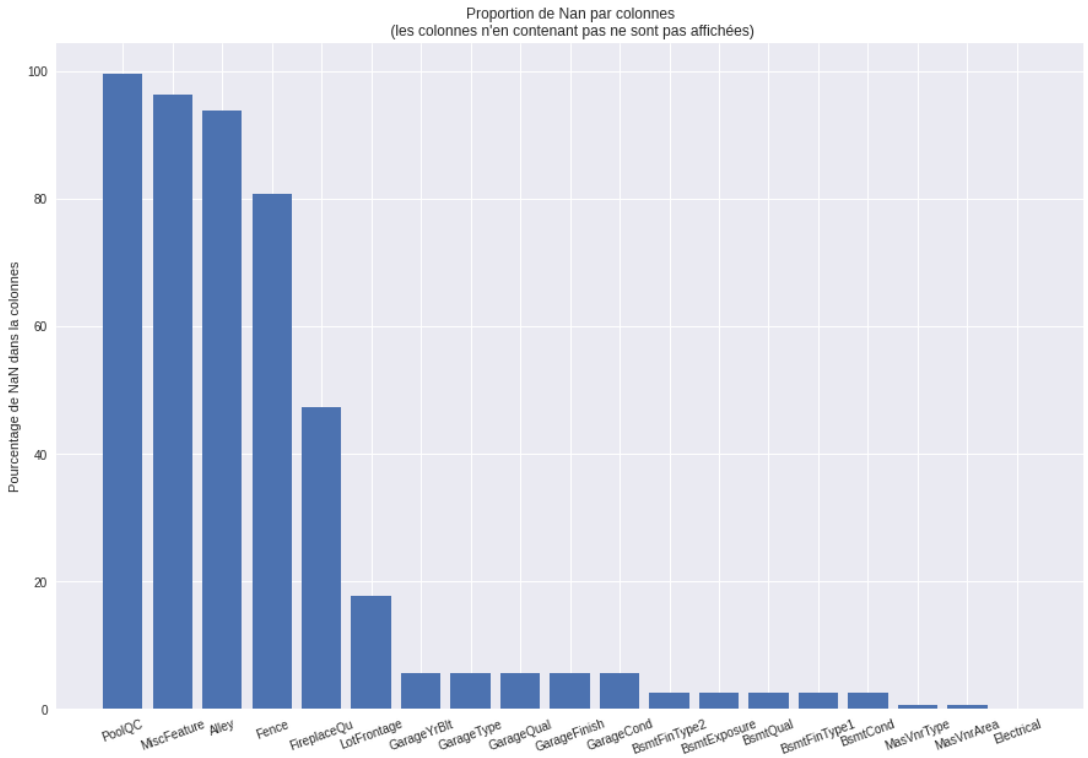
Le dataset contient des informations sur des maisons, sur lesquelles on veut utiliser des techniques de machine learning pour prédire leurs prix de vente (*SalePrice* dans le dataset). Commençons par étudier le jeux de données, il est composé de :

- 81 colonnes, les features
- 1461 individus

Les 81 colonnes représentent des informations, ou caractéristiques d'une maison, comme sa superficie, le type de matériaux utilisé, des notes attribuées pouvant décrire la qualité du garage ou des cheminées…

La première chose que nous avons du faire est de s'intéresser aux nombres de Nan (valeurs manquantes) dans le dataset.

Voilà un graphique représentant, en pourcentage, la proportion de NaN par colonnes.



Il nous faut donc d'abord “combler les trous". Pour cela, il nous à fallu regarder en détail à quoi chaque colonnes correspondait dans le fichier “data_description.txt” fournis avec la base de données.

On doit faire la différence entre les variable discrètes (qualitative) et continues (quantitative).

Par exemple, la colonnes *PoolQc* est une variable discrète, et correspond à :

PoolQC: Pool quality

Ex	Excellent
Gd	Good
TA	Average/Typical
Fa	Fair
NA	No Pool

Nous avons donc choisi dans ce cas de remplacer une valeur NaN dans cette colonnes par la chaîne de caractère “NA”, indiquant qu’il n’y a pas de piscine.

Pour les variable quantitative continues, on peut soit remplacer une valeur NaN par zéro, soir par la moyenne de cette variable.

Par exemple :

2ndFlrSF: Second floor square feet

Ici, il ferait plus sens de remplacer par zéro, NaN représentant probablement le fait qu’il n’y est pas de deuxième étage dans la maison.

Encodage

Une fois ce traitement effectué, il va falloir transformer toutes les valeurs littérales (les chaînes de caractères) en valeurs numériques, car la majorité des algorithmes de Machine Learning ne peuvent traiter que des valeurs numériques.

Là, on peut encore distinguer deux cas de figures, les variables ordonnées et celles non ordonnées.

Variable ordonnées	Variable non ordonnée
ExterQual: Evaluates the quality of the material on the exterior Ex Excellent Gd Good TA Average/Typical Fa Fair Po Poor	**MSZoning**: Identifies the general zoning classification of the sale. A Agriculture C Commercial FV Floating Village Residential I Industrial RH Residential High Density RL Residential Low Density RP Residential Low Density Park RM Residential Medium Density

Pour les valeurs ordonnées, nous avons manuellement transformer les données en valeurs numériques afin de conserver l'ordre présent, pour qu'une valeur faible indique une faible qualité et une valeur plus haute une meilleure qualité.

Pour les variables non ordonnées, nous avons tout d'abord utilisé un **Label Encoder**, qui permet d'attribuer automatiquement des nombre différents pour des valeurs différentes.

Cette méthodologie à le désavantage de créer un ordre artificiel entre les données, pour reprendre l'exemple de *MSZoning*, la zone d'habitations, il n'y a pas une zone qui est meilleure que les autres, cela est subjectif.

Pour pallier ce problème, nous avons donc également utilisé une autre méthode pour transformer les variables non ordonnées : **One Hot Encoder**.
Ici, on ne remplace plus les valeurs dans la colonnes, mais on crée de nouvelle colonnes à partir des valeurs que l'on souhaite encoder, pour *MSZoning* on aura alors une colonne *Agriculture*, une colonnes *Commercial* etc. On mettra alors des zéros et des uns en fonction dans ces colonnes pour indiquer l'appartenance ou non d'un indivis à cette zone.

Cette approche n'introduit donc pas d'ordre entre les données, mais à le désavantage de créer beaucoup de nouvelles colonnes.
A partir des 81 colonnes de base, on en obtient plus de 200.

Parallèlement à cela, nous avons également "fusionné" des colonnes entre elles en multipliant des colonnes indiquant la qualité et la quantité, car nous questionnions le fait d'avoir un 0, indiquant l'absence, dans la colonne de qualité, qui est ordonnée. Cela revenait à dire qu'un élément qui n'existe pas, à une faible qualité.
En fusionnant la quantité et la qualité, on obtient un indicateur, qui décrit en même temps ces deux valeurs, 0 indiquant l'absence.

En fonction de la façon d'encoder les données, nous avons créé plusieurs datasest à partir des données originales.

Réduction dimensions

Nous avons également extrait des sous ensemble de ces dataset afin de réduire le nombre de colonnes et "d'y voir plus clair".

Intuitivement, parmi les colonnes présentes, nous nous somme dit que certaines devaient avoir plus d'impact sur le prix de vente de la maison que d'autres.
Plus formellement, on a donc regardé la corrélation entre les variables aléatoire formée par les colonnes et la variable *SalePrice* (le prix des maisons).

On à donc calculé le coefficient de corrélation suivant : $$r = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y}$$

avec Cov(X,Y) la covariance et σX et σy l'écart-type des variables X et Y.

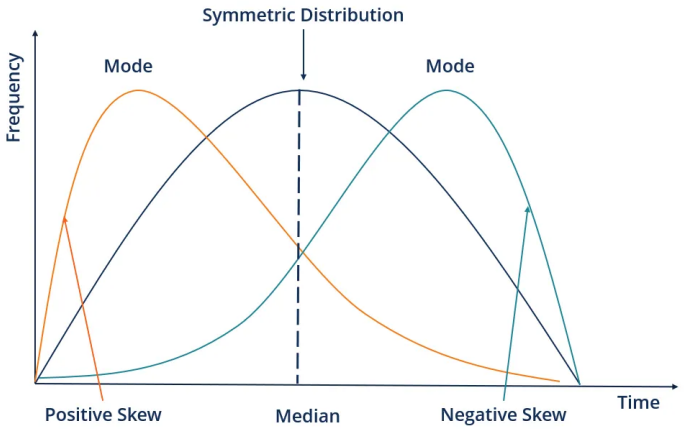
La valeur de **r** évolue dans [-1, 1] où r = 0 indique une absence de corrélation (ou une indépendance des variables) et **r** proche de 1 ou -1 une corrélation forte (positive ou négative).

Ainsi, on à extrait une vingtaine de colonnes dans les deux cas (Label Encoder et One Hot Encoder) pour lesquelles **r** > 0.4.

Une autre transformation des données que nous avons étudiée est la transformation des distributions des variables au sein du dataset.
Certains modèles de Machine Learning font l'hypothèse que les données sur lesquelles on réalise leur apprentissage forment une distribution normale. Or, dans la pratique, on a rarement des données qui suivent exactement une loi normale, et on voudrait donc "corriger" ces données, afin de "faciliter" l'apprentissage de nos modèles.

Skewness

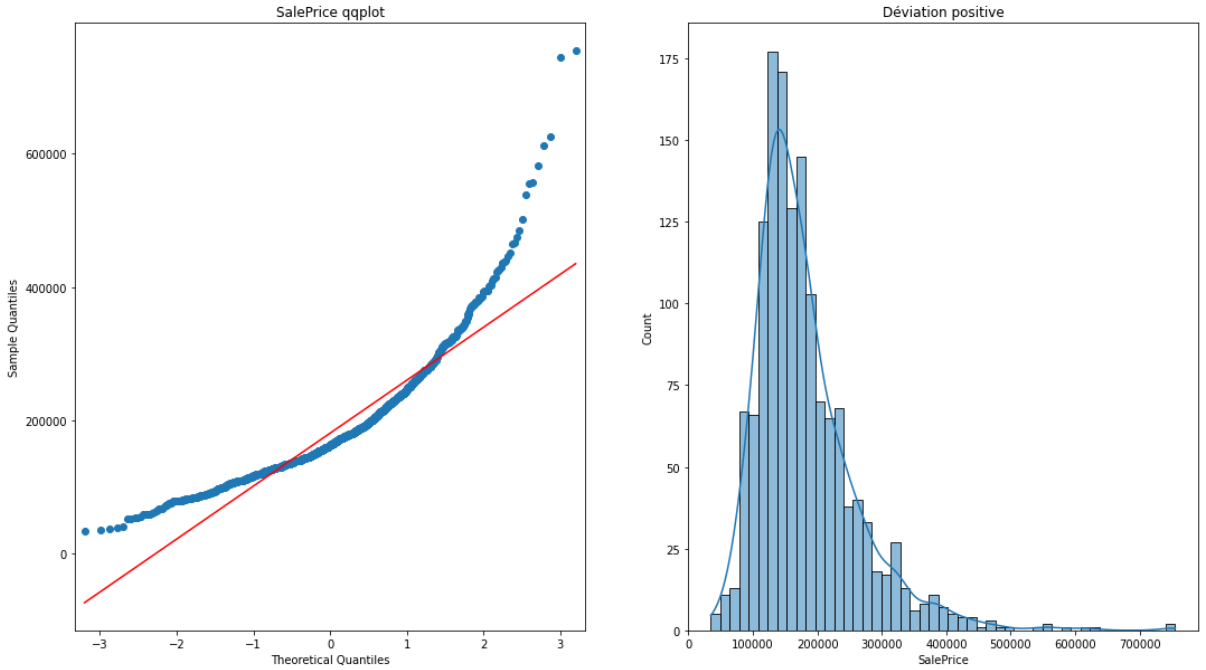
Il existe des manières d'étudier la dérive de la distribution d'une variable à la loi normale. On parlera de "Skewness" en anglais.



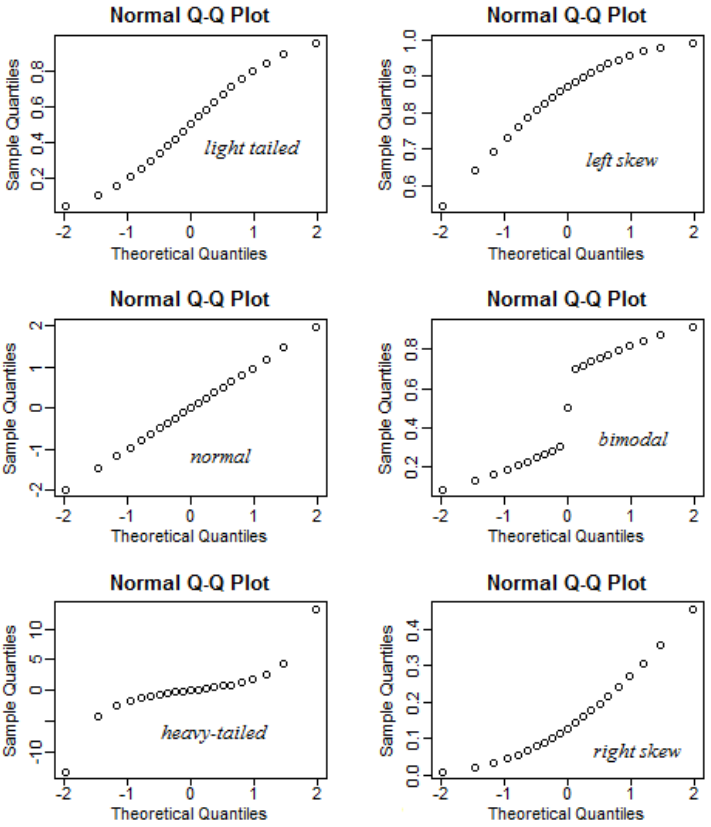
https://corporatefinanceinstitute.com/resources/knowledge/other/skewness/

La déviation (skewness) peut être positive ou négative, en fonction de où se trouve la queu de la distribution.

On peut tracer un graphique appelé **qqplot** qui permet de visualiser comment notre distribution dévie.

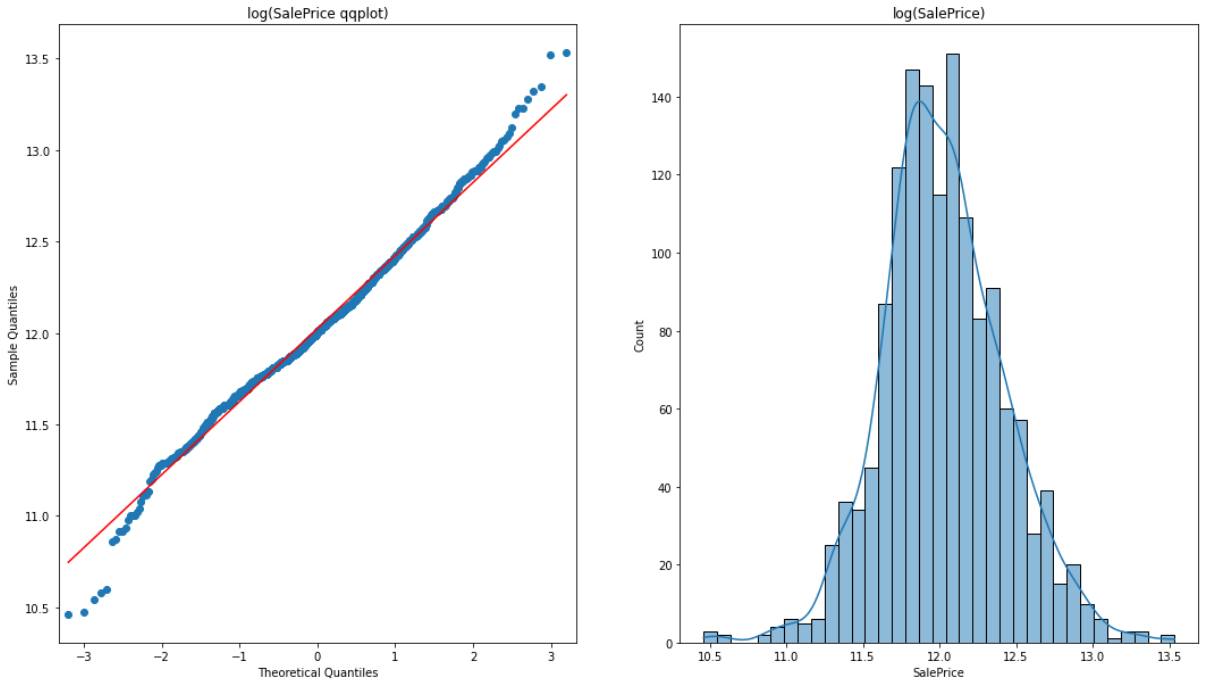


Grâce au qqplot on pourra alors récupérer des information sur la distribution de nos données :



https://stats.stackexchange.com/questions/101274/how-to-interpret-a-qq-plot

On pourra alors “corriger” notre distribution en appliquant le logarithme sur notre variables :



De plus, comme nous le verrons dans la partie sur la régression, il peut être intéressant de prendre le logarithme de la colonne *SalePrice*, (qui est la colonne target) car cela aura pour effet, à de **pénaliser autant les erreurs** de nos modèles **sur les petits prix que sur les grands** lors de l'apprentissage.

Classification

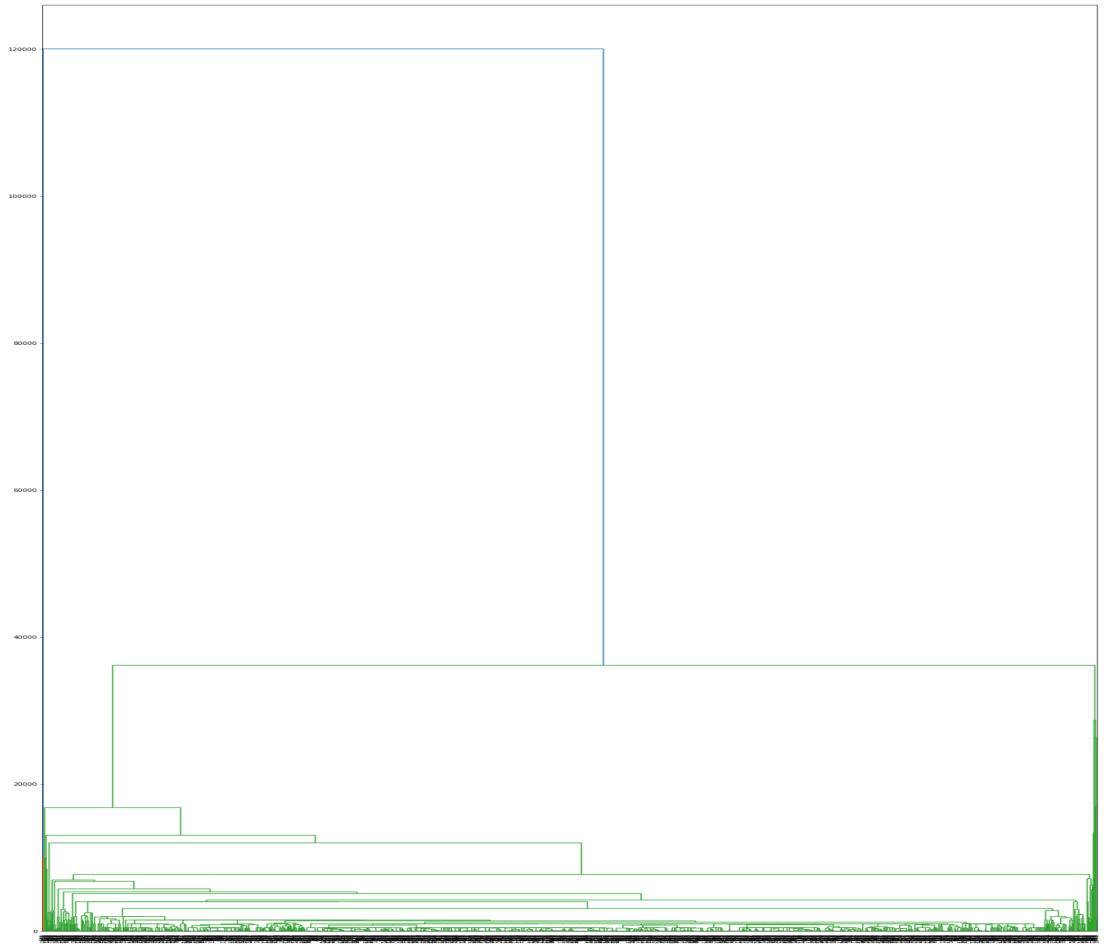
Clustering

La deuxième étape du projet était de faire de la classification. Nous avons donc commencé par chercher comment découper les tranches de prix des maisons pour créer des classes appropriées. Nous nous sommes naturellement dirigés vers des techniques de clusterings qui nous semblaient être une bonne solution à notre problème.

En effet, le clusterings à pour but de faire une analyse statistique des données et de regrouper celles présentant des propriétés similaires. Les deux algorithmes que nous avons appris cette année sont le clustering hiérarchique et les k-moyennes.

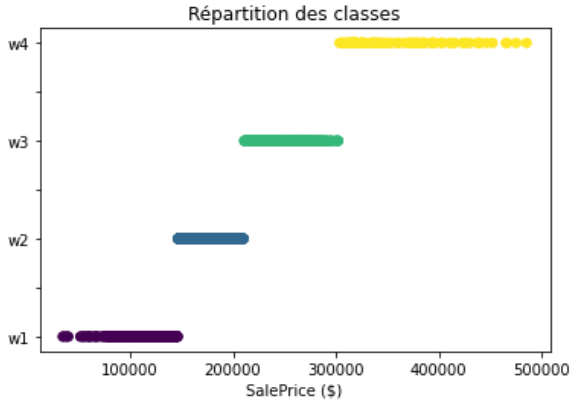
Nous avons dans un premier temps essayé la première technique en affichant le dendrogramme des prix. On peut voir que cette technique marche mal pour notre problème, en effet, plusieurs maisons du datasets possèdent le même prix ce qui explique la grosse accumulation de clusters dans la partie gauche du graphique.

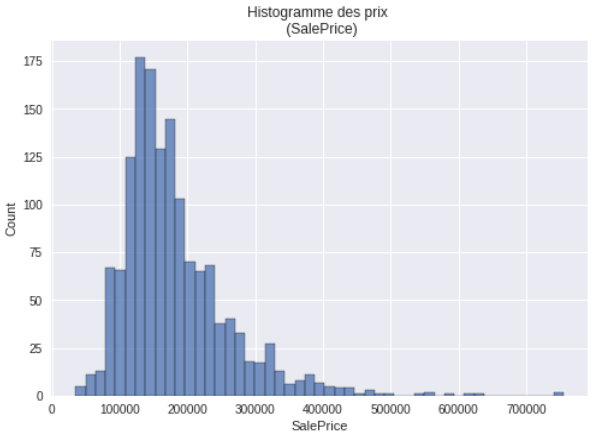
Dendrogramme des prix



Nous nous sommes donc dirigé vers la méthode des k-moyenne qui à l'avantage de pouvoir choisir le nombre de clusters à créer. Pour trouver la meilleure, nous avons testé plusieurs valeurs de k et avons fini par décidé de k = 4.

Comme on peut le voir sur ces graphiques, les classes donnent une bonne représentation des prix:
- W1 les maisons peu chers (la queue gauche de l'histogramme entre 0 à 10 000)
- W2 les maisons de classe moyenne (le pic de l'histogramme)
- W3 les maisons haut de gamme (la pente descendante de l'histogramme entre 220 000 et 300 000)
- W4 les maisons de luxe (la queue droite de l'histogramme)





Méthodologie

Après avoir établi les classes nous avons commencé à tester les algorithmes de classification, pour cela nous avons mis en place la méthodologie suivante:

1. Choisir quelques classifieurs qui sembles adaptés
2. Déterminer les valeurs de leurs hyperparamètre par cross validation
3. Choisir celui qui donne les meilleurs performances sur la base d'apprentissage (valeur par cross validation)
4. Tester ce modèle sur l'ensemble de test.

Classifieur

Nous avons testé les classifieurs sur les quatres datasets créés lors du prétraitement, pour cela chacun des datasets ont été découpés en deux parties, 75 % des données nous ont servi de base d'entraînement pour nos modèles, et le reste, a évaluer le meilleurs.

Classifieur	Score	Dataset
9-PPV	0.711	LE20
Gaussian	0.723	LE20
Parzen*	-4906251300	X
MLP	0.688	LE20

* Le classifieur par la méthode de Parzen ne marche pas, le dataset possède des valeurs discrètes qui ne sont pas adaptées à cet algorithme.

Le modèle que l'on a retenu, parmi tout ceux vu en cours, est le classifieur Gaussien .
Son score sur la base de test est de 76% de taux de bonne classification.

Régression

RMLSE

Nous nous sommes enfin tournés vers la régression. Pour la sélection de nos modèles, nous avons utilisé la même méthodologie que précédemment. En revanche, nous avons changé la métrique utilisée pour évaluer les performances de nos modèles pour utiliser celle en vigueur sur Kaggle : **R**oot **M**eans **S**quared **L**ogarithmic **E**rror (**RMSLE**).

RMLSE : $$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(x_i+1) - \log(y_i+1))^2}$$

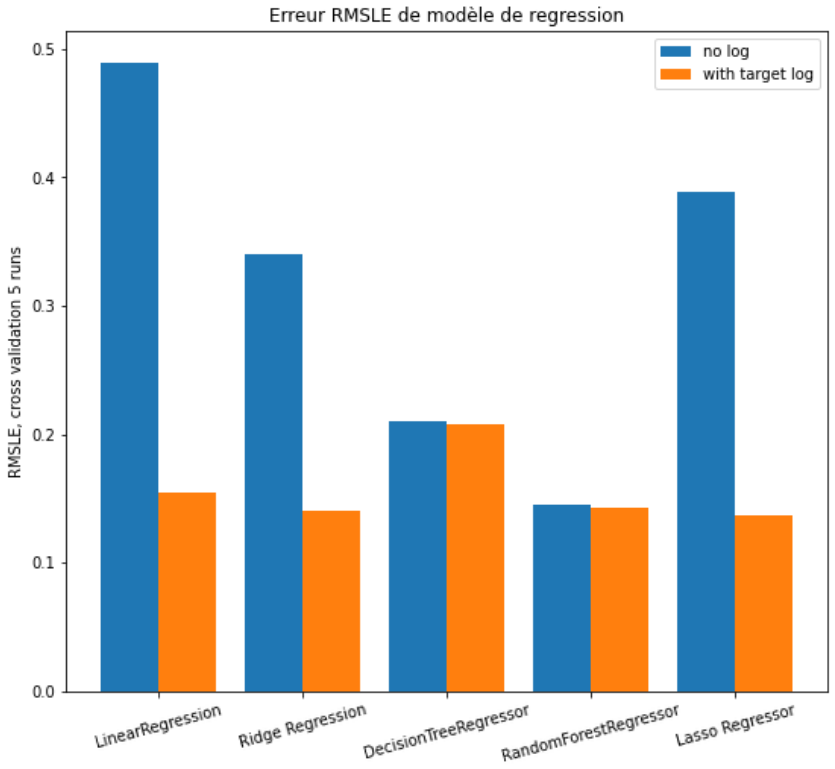
Avec x_i les prédictions et y_i les valeurs réelles

Cette métrique possède des caractéristiques intéressante :

- Elle est robustes aux valeurs extrême (outliers)
- Elle mesure l'erreur relative, soit x une prédiction et y la valeur réels,
  alors, si x = 90 et y = 100 alors l'erreur E vaut 0.1053, si x = 9000 et y = 10000, alors E vaut également 0.1053
- Enfin, elle pénalise plus fortement les sous-estimations que les sur-estimations (utile en finance mais pas dans notre cas).

Comme évoqué dans la partie sur le prétraitement, transformer la colonne de target par le logarithme permet d'augmenter les performances des modèles, car nos modèles apprendrons à prédire correctement aussi bien les petits prix que les plus grands. Sans cette transformation, les plus grands prix donnant plus de pénalité, le modèle “apprendrai à prédire le prix des maisons les plus chères”.

Voilà un graphique illustrant ce phénomène, et les performances par cross validations (5 folds) obtenus sur l’espace d'apprentissage après avoir “tuner” les modèles par cross validation :

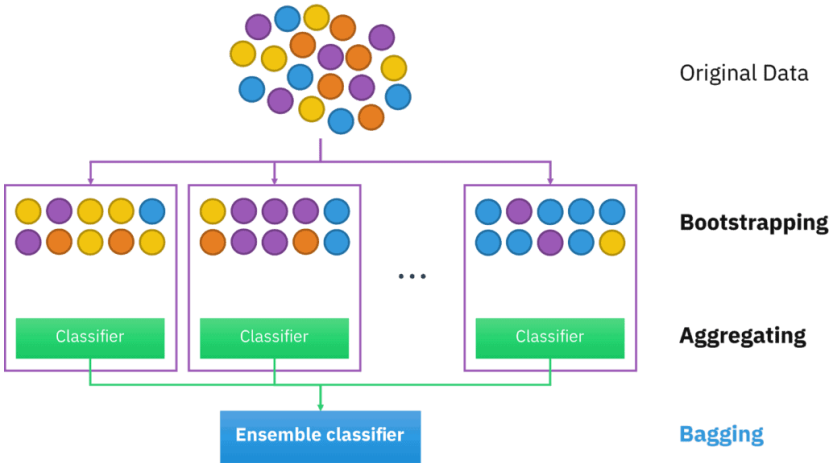


Régresseur	Score RMSLE	Dataset
Linear Regression	0.154	LogOHE
Ridge	0.143	LogOHE
Lasso	0.136	LogOHE
RandomForestRegressor	0.141	LogOHE

On retiendra deux modèles, le RandomForest Regressor et le modèle de régression linéaire Lasso.

RandomForestRegressor

Ce modèle est un modèle dit de **Ensemble Learning**, plus précisément de **Bagging**.
Le principe d'une telle approche est d'utiliser plusieurs modèles de régression, chacun entraîné sur une partition du dataset et de prendre pour sortie une combinaison des sorties de ces modèles internes :



https://cnvrg.io/random-forest-regression/

Ce type de modèle permet de limiter le sur-apprentissage.

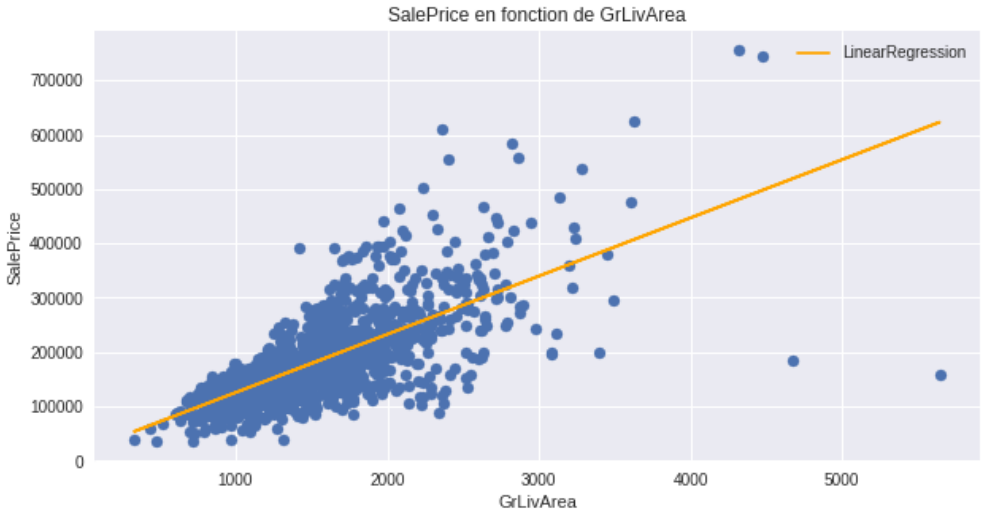
Dans notre cas, les modèles internes sont des arbres de décisions, de paramètres différents.
Les hyperparamètres déterminés par cross validation sont les suivants :

- n_estimators = 90
- max_depth = 330
- min_samples_leaf = 2

Il nous a donné une erreur RMSLE = **0.14274** sur le jeu de test.

LASSO

Parlons d'abord de la régression linéaire, le but de celle-ci est simplement de tracer une droite décrivant un nuage de points. Le fait que LASSO donne de bonnes performances sur le dataset étudié peut venir du fait qu'il contient des colonnes sur lesquelles la régression linéaire est très appropriée. Comme par exemple la colonne *GrLivArea* :



Le modèle LASSO est une version "améliorée" de la régression linéaire simple.
Sans rentrer dans les détails, trop techniques pour nous, on notera qu'il détermine des coefficient pour chaque colonne du dataset en minimisant une fonction par coordinate descente.

Pour ce qui est des hyperparamètres, LASSO n'en possède qu'un seul, qui est alpha, pour lequel une cross validation nous à donnée alpha = 0.0005591836734693878

L'erreur obtenue sur le jeu de test s'élève à **0.12945** (RMSLE).

Conclusion

Nous avons remarqué que nos classifieurs donnent de meilleurs résultats sur les datasets obtenus avec un Label Encoder et pour lesquelles on a réduit le nombre de colonnes.

En revanche, pour la régression, les modèles donnent de meilleurs résultats sur les datasets obtenus avec One Hot Encodeur et dont on n'a pas réduit la dimension.

Lors de ce projet, nous avons pu appliquer des notions vues au cours des semestres précédents, des notions d'Apprentissage Automatique, avec la cross validation, en faisant attention à ne pas sur-apprendre, les arbres de décisions, mais aussi des notions de Probabilités et Statistique comme le clustering.

Nous avons pu également découvrir de nouvelles techniques et problématiques liées à la régression et à l'apprentissage comme la régression linéaire ou l'Ensemble Learning.

Ce projet nous a également familiarisé avec la plateforme Kaggle, où nous avons pu participer à la compétition en soumettant nos prédictions.