L2 mini projet
challenge “friend”
Membre : Hamza Baa (groupe 2), Yacine Bouhadda(groupe 2), Simon bibitchkov(groupe 3bis), Thomas Annino (groupe 2bis), Marvin Comlan (groupe 3), Samuel DIOP (groupe 3)
URL challenge : https://codalab.lri.fr/competitions/112#participate
URL Github : https://github.com/Slownite/mini_projet
URL video : https://www.youtube.com/watch?v=OHZSAT5GiL8&feature=youtube
numéro code soumission : 4353
présentation:https://drive.google.com/open?id=1CyLBj6xRwPTCSwrM4hsBqwa2fQ5O7Q2Z

Projet Amitie

➢Context et description du projet

Le challenge “friend” que nous avons choisi à pour but de prédire le prix optimal d’une maison: c’est un problème de type régression. Nous utiliserons pour cela le dataset des maisons vendus entre mai 2014 à mai 2015 par King County à Seattle. Soit 21613 données. Nous avons accès à 18 labels différents allant du nombre de salle de bains à la surface totale. Afin de parvenir à nos fins nous divisons notre projet en 3 parties, une de preprocessing pour nettoyer nos données et rendre la prédiction plus performante, une de prédiction entraînant la prédiction à l’aide d’un modèle choisi, et un groupe visualisation qui créera une interface graphique pour mieux visualiser les datas et les résultats. Ce qui nous a poussé à choisir ce projet.

1. preprocessing :

Après le grand travail fait par les Master ‘recueil et division de données’, vient le tour à la phase de preprocessing.

Introduction et problématique:
le jeu de données brutes pose quelques problèmes qui doivent être résolus avant de passer aux étapes suivantes ,et pour chaque problème une solution est proposée par divers méthodes.

Définition : c’est un ensemble de transformations sur le jeu de données dans le but de les préparer à l’étape qui suit (entraînement et validation du modèle),c’est une étape primordiale puisqu'elle participe non seulement à l’amélioration des résultats et de la puissance prédictive de notre algorithme machine learning ,mais aussi à l’assurance de la bonne exécution des algorithmes choisis pour l’entraînement et pour la cross-validation ,

A. normalisation :

pourquoi la normalisation ?

il arrive souvent que les variables (features) initiales évoluent dans des proportions qui sont très différentes.Cette différence d’échelle engendre des problèmes pour certains

algorithmes (algorithme de recherche du vecteur des paramètres comme celui de la descente du gradient);il s'agit de l'influence sur la *qualité et la fiabilité des résultats* ,l*a complexité* des ses algorithmes et même parfois *la divergence* de ces derniers.
donc *la normalisation* constitue une solution pour ce problème ,et s'agit de mettre les variables à la même échelle.

Exemple :la superficie et le nombre de pièces.

motivations :

- ★ *plusieurs algorithmes de normalisation existent ,dits «scaler» en anglais. l'API scikitlearn de python offre plusieurs 'scaler' dans son module preprocessing citant comme exemple standarScaler, MinMaxScaler et Normalizer.*

- ★ *Détail de MinMaxScaler :ce scaler en principe associe à chaque valeur de X (une donnée brute) une valeur Xstd (X normalisée) de telle sorte que Xstd=X-min(X)/max(X)-min(X) .*

- ★ *parfois c'est très important le choix du scaler (introduction des notions probabilistes,nature des variables ...etc),dans uns régression linéaire multivariée (à plusieurs variables) on peut se contenter de travail à l'aise avec les deux premiers cités en exemple ,mais rien n'empêche de voir le Normalizer .*

Les deux figures suivantes illustrent l'échantillon normalisé à l'aide de deux scaler différents

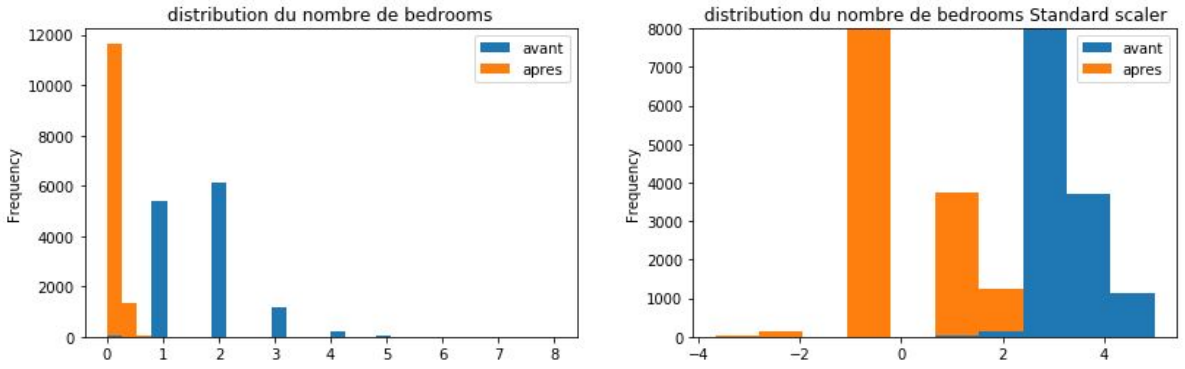


figure 2.2 *: distribution de la variable bedrooms avant et après la normalisation*

B. Les espaces de grande dimension :

Introduction et problématique:

il arrive souvent que nous travaillons sur des espaces de grande dimension (correspond au nombre n de caractéristiques ou variables prise pour chaque donnée)ce qui augmente la probabilité de tomber dans ce qui est dit sur-apprentissage ou 'overfitting' en anglais ,et qui dit sur-apprentissage dit impossibilité de généralisation d'un modèle,donc notre modèle est à rejeter.

Méthodes de résolution

Deux approches sont envisageables pour résoudre ces problèmes. Ces dernières ont pour but la réduction de notre espace de variables et cherchent à trouver un sous-espace de feature plus pertinent pour améliorer la modélisation:

- sélectionner un nombre restreint de variables les plus importantes:c'est une approche machine learning ; elle se sert des algorithmes du machine learning pour sélectionner et classifier nos variables par ordre d'importance .

- Création de variables 'synthétiques' à partir des variables initiales:dans ce cas on ne travaille plus avec nos variables initiales mais avec des nouvelles variables créées en manipulant les initiales(combinaisons linéaires ou non linéaires des variables) ,ce qui nous permet de construire un espace de dimension réduite .

pour nous on a essayé les deux ,et finalement nous avons travaillé avec la deuxième approche en choisissant l'analyse des composantes principales (PCA) comme algorithme pour la projection des variable de départ sur un hyperplan réduit.
principe de fonctionnement de la méthode PCA est illustré dans la figure suivante:

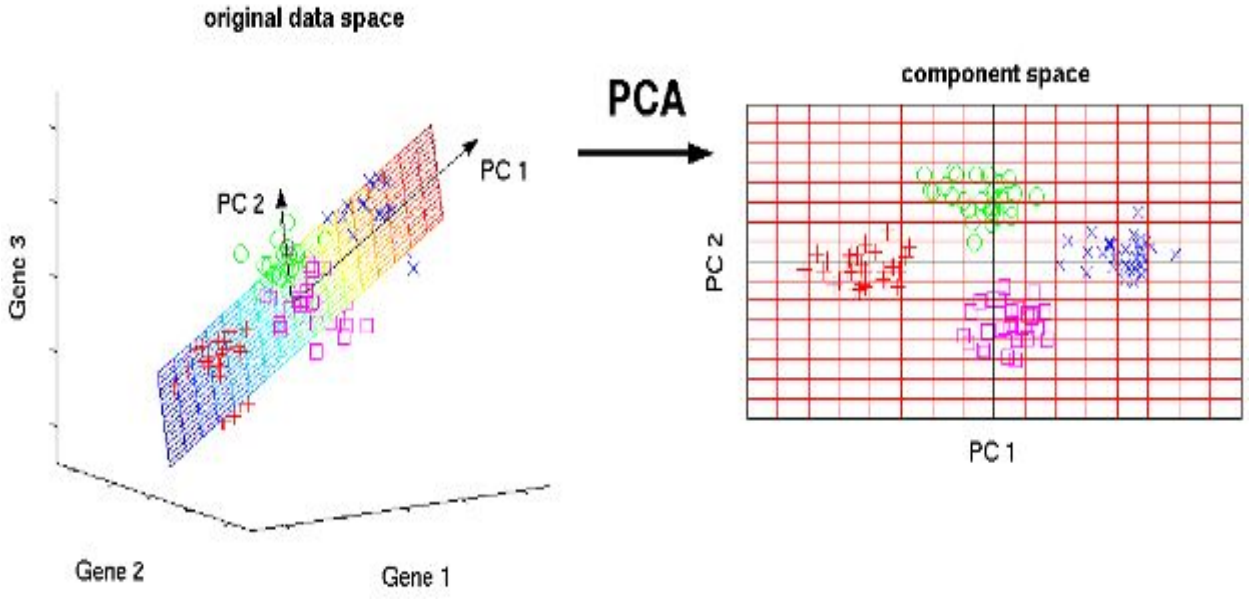


figure 2.3 :*projection d'un jeu de données en 3D sur un espace 2D*

motivations :

il est à noter que les espaces de grandes dimensions engendrent d'autres problèmes à part celui de l'overfitting à titre d'exemple celui des données éparses comme montre la figure qui suit.
plusieurs méthodes existent pour implémenter chacune des deux approches précédentes ; *la régression pas à pas* qui est une approche statistique est moins performante pour la

sélection des variables et *l'analyse en composantes principales* pour la création des variables synthétiques.

C. Valeurs manquantes et valeurs aberrantes:

lorsqu'on travaille sur des données réelles ,on est très souvent amené à considérer des valeurs manquante ou aberrantes.

1. *Valeurs manquantes:* réparties en deux catégories
   - ➢ lorsque l'on dispose d'aucune information sur l'individu (c'est à dire une ligne complètement vide)dans ce cas on parle de non-réponse totale.

   - ➢ Lorsque l'on dispose d'une information incomplète (une ligne partiellement enregistrée) ,là on parle de non-réponse partielle.

comment traiter les valeurs manquantes?
beaucoup de méthodes existent pour cela ,ceci peut être justifié par les *différentes natures des données manquantes* ,mais dans la plupart des cas on se contente d'utiliser la plus simple,qui est :
l'analyse des données complètes qui consiste à ne conserver que les observation qui ne contiennent aucune donnée manquante.
En revanche on peut bien remplacer ces valeurs manquante par des valeurs plausible en se servant la technique d'*imputation* **.**

2. *Valeurs aberrantes :*

les données peuvent aussi comprendre des valeurs aberrantes ou extrêmes.

comment traiter les valeurs aberrantes ?
Il faut être très attentif à ce type de problème car les valeurs aberrantes peuvent ne pas être des erreurs (i,e elle représente un comportement réel ou des événements rares);dans ce cas il convient d'en tenir compte dans les analyses ,et si elles correspondent à des erreur on peut bien les traiter avec des méthodes de substitution et les considérer comme valeurs manquante.

comment repérer et récupérer les valeurs aberrantes?
Aussi comme les méthodes de résolution des problématiques précédentes divers méthodes existe pour repérer les valeurs aberrantes.la plus connue étant sans doute celle de la boîte à moustache 'boxplot' en anglais.

la **figure 2.4** illustre la représentation de nos variables à l'aide des boîtes à moustaches

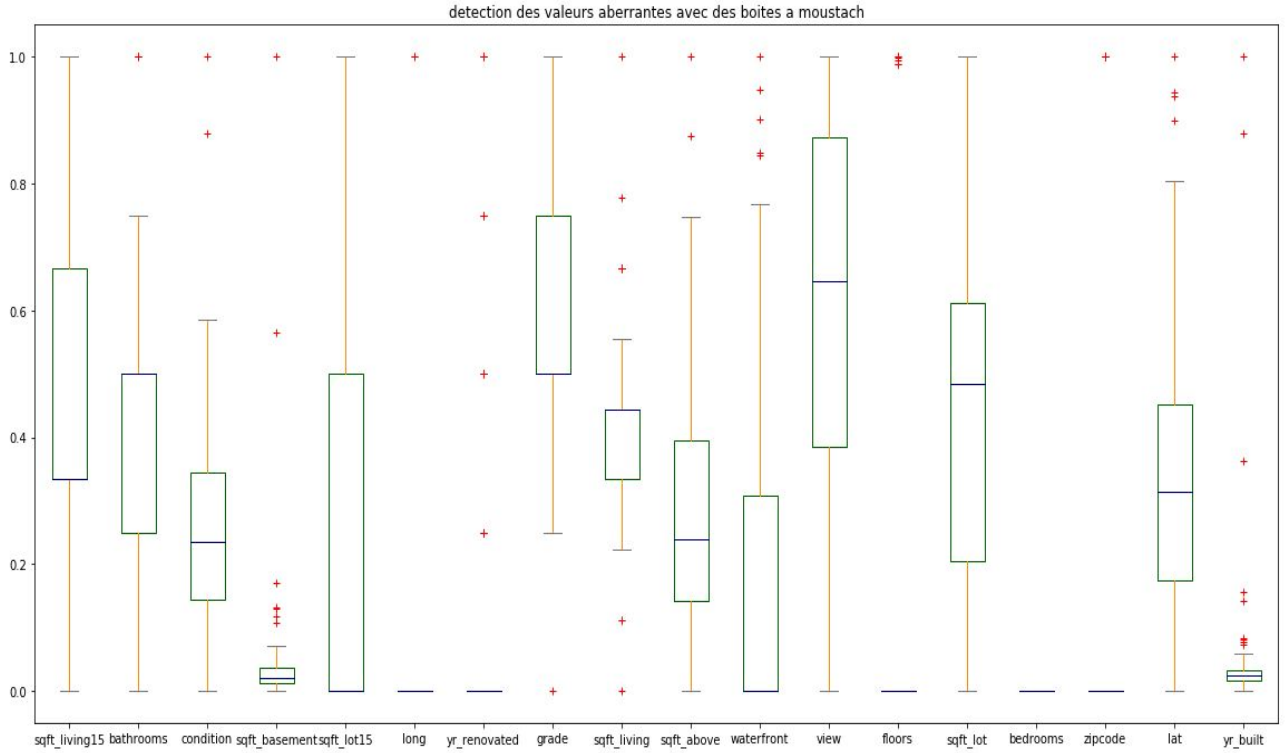


figure 2.4 : *détection des valeurs aberrantes à l'aide des boxplot*

motivation : comment utiliser la boîte à moustache pour repérer des valeurs aberrante ?

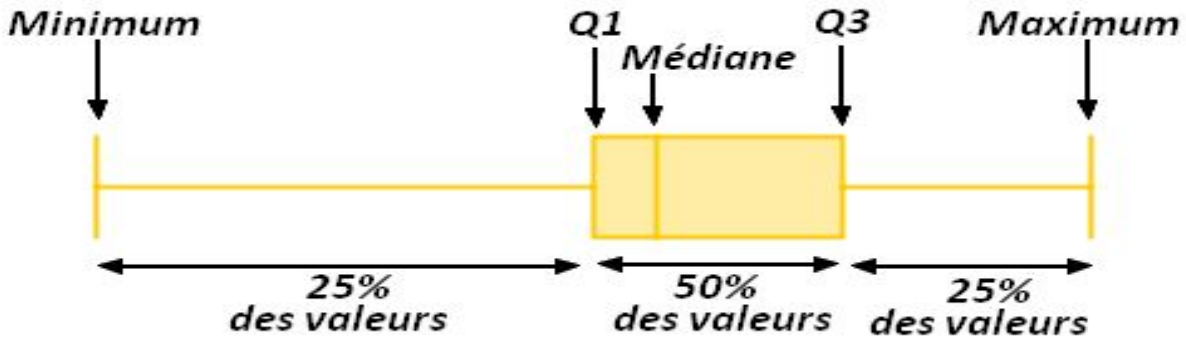


figure 3.1 : montre le principe générale de la représentation d'une population à l'aide d'une boîte à moustache.

explication: après la représentation de notre variable statistique à l'aide d'une boîte à moustache (calcule la médiane ,le premier quartile Q1 et le troisième quartile Q3) on cherche à déterminer les moustaches de notre boite (Minimum et Maximum) de la manière suivante :

on calcule :

L'écart Interquartile=Q3-Q1 puis

Minimum=Q1-1.5* L'écart Interquartile

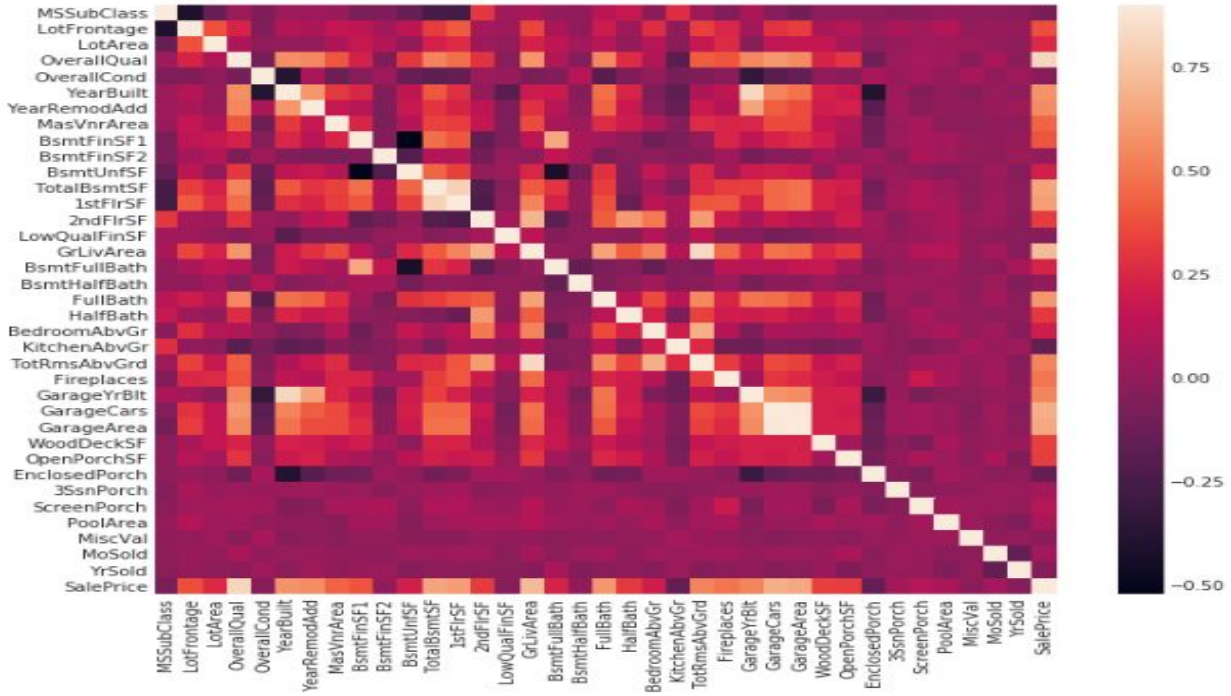
Maximum=Q3+1.5* L'écart Interquartile

★ on dit qu'une valeur V de la population est aberrante si V∉[**Minimum , Maximum**]

2. Visualisation:

Le but de notre groupe est de rendre les données lisibles et facilement interprétables par le reste du groupe, pour qu'ils puissent trier les données de façon efficace en ne gardant que celles qui sont pertinentes. Pour cela nous utiliserons différents types de graphiques, avec différentes variables.

Nous avons commencé par faire une heatmap pour montrer la relation entre les prix et les variables, ce qui nous a permis de sélectionner les variables fortement reliées entre elles.



Suite à ces analyses préliminaires, on a essayé d'avoir le plus grand nombre de données possibles de façon efficace, on utilise donc un affichage de multiples graphiques :

Ce travail nous as permis de mettre en valeur les corrélations les plus importantes entre différentes variables, afin de préciser les directions à suivre dans le model d'apprentissage.

Les autres membres du groupe ont eu accès à notre code et graphiques par le biais d'une classe avec une documentation ils ont pu faire appel aux méthodes fournies pour avoir une visualisation des données.

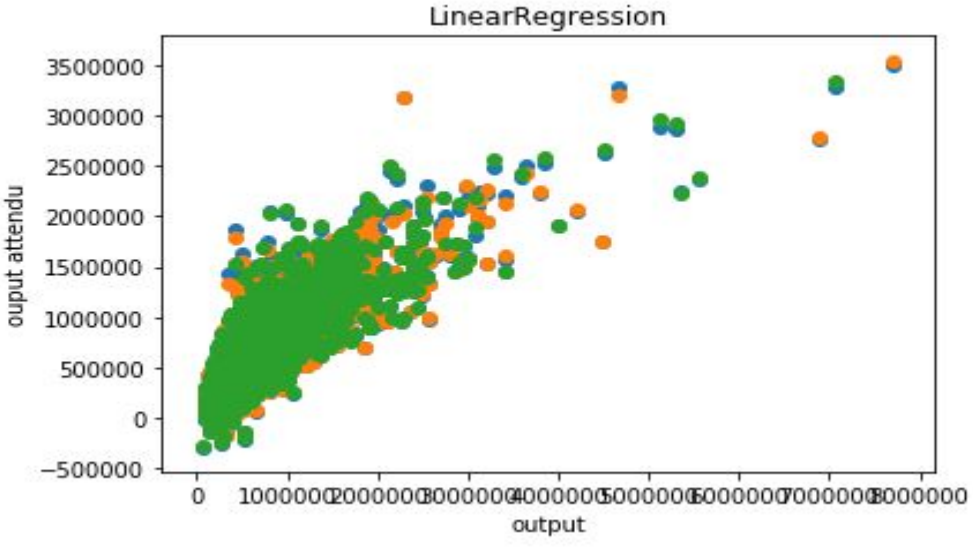
Chaque méthode de notre classe as été testée avec une courte liste de valeurs ce qui nous as permis de nous assurer qu'elle affiche des représentations cohérentes. Nous n'avons pas vraiment pu faire d'assert étant donné que les résultats que nous obtenions étaient visuels, il as donc fallu faire ce travail de vérification manuellement, ce qui explique que nous n'ayons testés que quelques valeurs.

L'une des difficultés que nous avons rencontrés était le mauvais tri des données dans le dataset, les données n'étaient en fait pas triées correctement, ce qui nous as ralentis dans notre travail puisqu'il a fallu retrouver quelle donnée était associée à quelle variable.

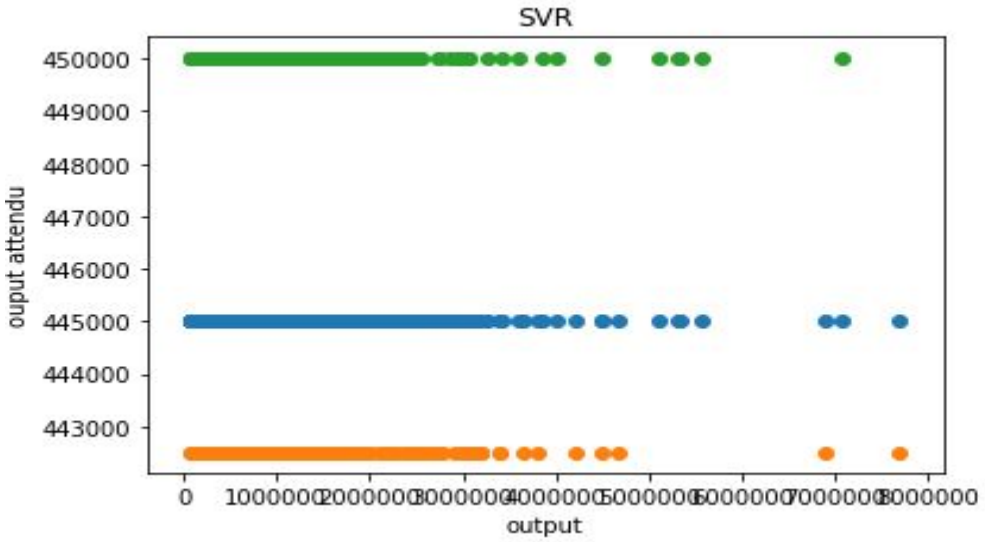
3. Model :

L'objectif du groupe "model" fut de choisir et améliorer le modèle prédictif. Pour cela nous avons commencé par nous intéresser aux différents modèles existant.

Tout d'abord LinearRegression:

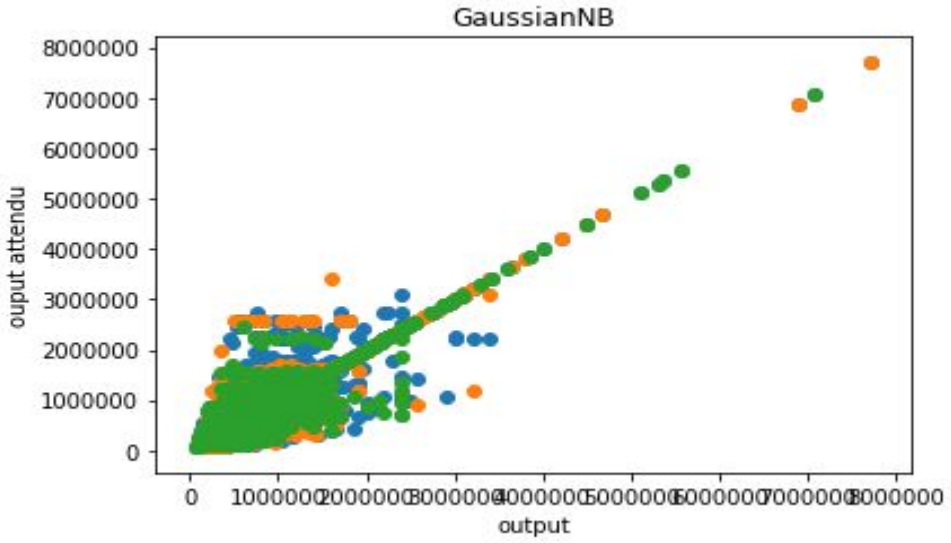


La distribution des points était plutôt bonne(une Cross-validation de 0,655). Mais ce n'était pas parfait. En effet presque toutes les valeurs se trouvaient trop haut par rapport à la réalité. Donc notre premier objectif fut de battre ce record grâce à un autre model. Après recherches approfondies notre choix s'est porté sur SVM.SVR



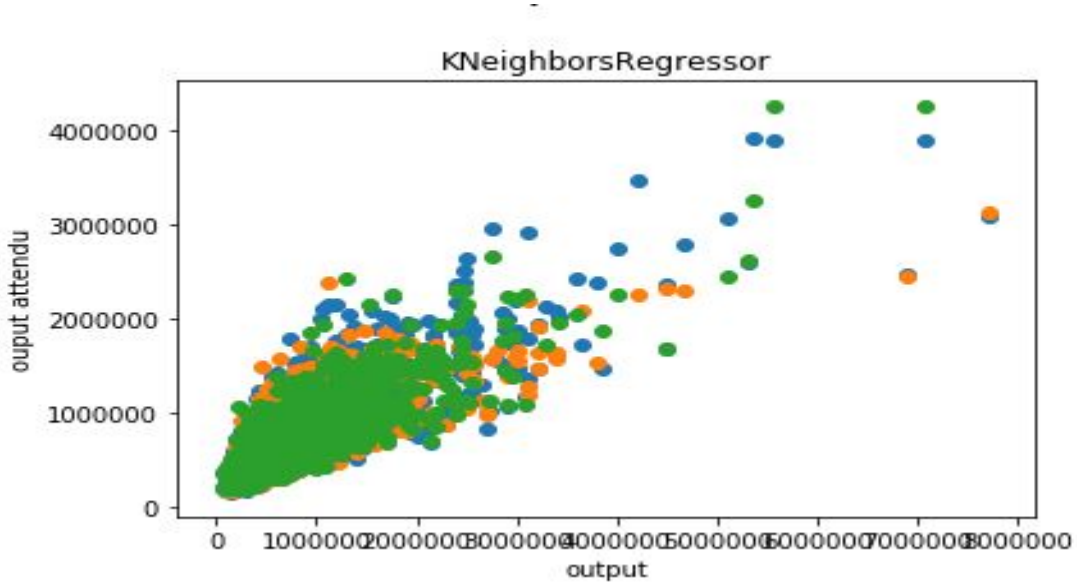
Mais comme vous pouvez le constater sur la figure ci-dessus les points sont disséminés entre trois segments de valeurs. Le score de validation étant donc moins bon que le hasard (score de -0,054). Nous avons essayé de voir si les hyperparamètres pouvaient y changer quelque chose mais rien n'y fait.

Ce qui nous amené à tester d'autre modèles dans leur version "regressor" proposer dans la tableau bonus. Ce fut au tour de GaussianNB.



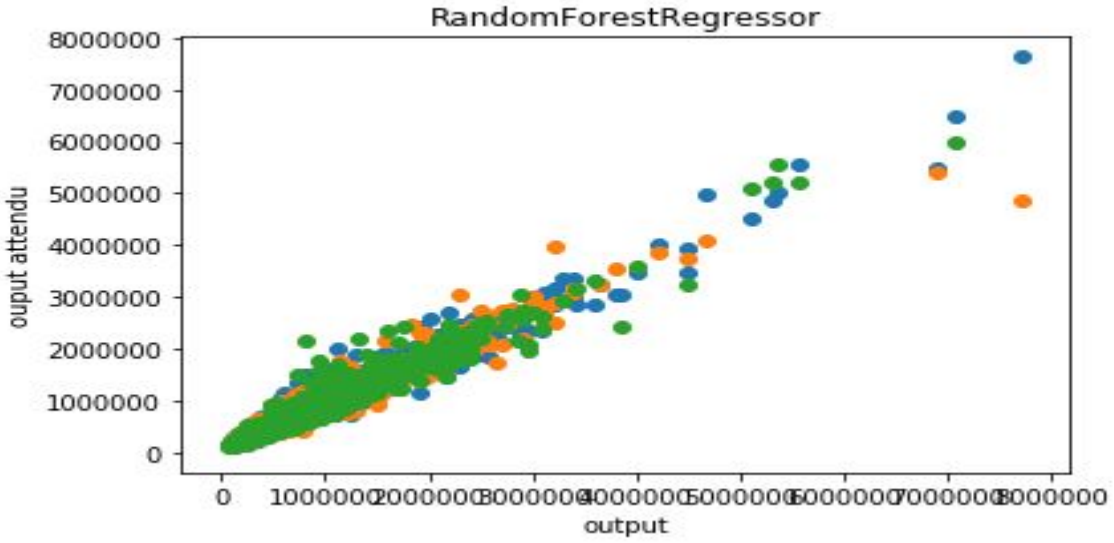
Avec un score de cross Validation de 0,336. Nous avons préféré ne pas nous étendre sur ce model.

Nous sommes donc passés sur KNeighbors.



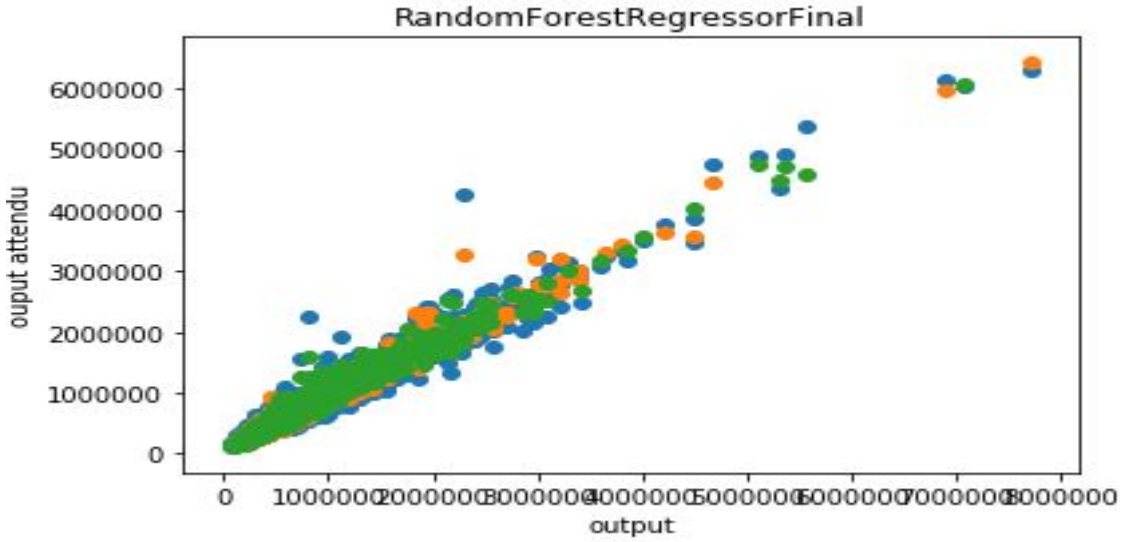
Ce modèle était plus performant mais pas autant que LinearRegression beaucoup trop de maisons étaient surévaluée ou sous évaluée.

Ce qui nous fit arriver au modèle choisi en définitive RandomForestRegressor.



Avec une cross-Validation de 0,771 ce fut le meilleur modèle rencontrer. Comme vous pouvez le voir sur le graphe les points sont répartis autour de la droite, avec une marge d'erreur assez faible. Ce qui nous poussa à créer un algorithme jouant sur les

hyperparamètres. Tout d'abord nous avons joué sur le nombre d'arbre. Après avoir tester de 1 à 100 arbres la valeur permettant le meilleur résultat possible fut 72. Ce qui nous amène à un score de cross-validation de 0.7903. Par la suite on a fait varier "min_sample_leaf" et "max_features". Le premier de 1 à 11 et le deuxième pour les valeurs 0.3, 0.6, 0.9, 'auto'. Le choix de ces valeurs fut par tâtonnement et en fonction de la puissance de calcul à notre disposition. Notre Algorithme en définitive à sélectionner 3 pour "min_sample_leaf" et 0.6 pour "max_features" ce qui nous donne sur Codalab un score de 0,80.



Ce graphe nous montre bien l'impact de notre algorithme sur le modèle, les points étant encore plus agglomérés autours des "output_attendu".

➢CONCLUSION :

Nous pourrions dire que le challenge fut une grande source d'apprentissage pour tout le monde. Il fut par moment stressant, par moment énervant mais surtout instructeur sur une multitudes de points: la technologie, l'organisation, le travaille de groupe.

Si nous avions un conseil à donner au L2 de l'année prochaine ce serait de prendre le temps de bien comprendre le sujet, de ne pas hésiter à passer des heures sur google pour être sûr d'avoir bien compris. Ce fut notre erreur et je pense que plus aucun d'entre nous la commettra.

Pages Bonus

Qu’est ce que le sur-apprentissage ?

Le sur apprentissage est en général provoqué par un mauvais dimensionnement de la structure utilisée pour classifier. A cause de sa grande capacité de stockage d'information, une structure subissant le surapprentissage ne pourra plus généraliser les caractéristiques des données : les prédictions sur les nouveaux échantillons sont donc compromis.

Qu’est ce que la cross-validation ?

C'est une méthode qui, à l'aide d'une technique d'échantillonnage, estime la fiabilité et prédit l'efficacité d'un modèle sur un ensemble de validation hypothétique lorsque un ensemble de validation indépendant et "clair" fait défaut.
Il y en existe de plusieurs types mais nous en avons utilisé en particulier le k-fold cross-validation

(citation wikipédia):
on divise l'échantillon original en k échantillons, puis on sélectionne un des k échantillons comme ensemble de validation et les k-1 autres échantillons constitueront l'ensemble d'apprentissage. On calcule comme dans la première méthode le score de performance, puis on répète l'opération en sélectionnant un autre échantillon de validation parmi les k-1 échantillons qui n'ont pas encore été utilisés pour la validation du modèle. L'opération se répète ainsi k fois pour qu'en fin de compte chaque sous-échantillon ait été utilisé exactement une fois comme ensemble de validation. La moyenne des k erreurs quadratiques moyennes est enfin calculée pour estimer l'erreur de prédiction.

Le tableau des scores :

Method	NaiveBayes or Gaussian classifier	Linear regression	Decision Tree	Random Forest	Nearest Neighbors	SVR
Training	0.6529490617617095	0.657554582874188 1	0.99905142178045 94	0.95893952073463 17	0.669807542315359	
CV	0.3781591030801296	0.653456319035826 6	0.57358587732603 52	0.77216289574728 64	0.461342112594392 2	
Valid(ation)	0.4593	0.6638		0.8011		

NaiveBayes or Gaussian classifier

La classification naïve bayésienne est un type de classification Bayésienne probabiliste simple basée sur le théorème de Bayes avec une forte indépendance (dite naïve) des hypothèses. Elle met en œuvre un classifieur bayésien naïf, ou classifieur naïf de Bayes, appartenant à la famille des classifieurs linéaires.

Linear regression

un modèle de régression linéaire est un modèle de régression qui cherche à établir une relation linéaire entre une variable, dite expliquée, et une ou plusieurs variables, dites explicatives.

Decision Tree

L’apprentissage par arbre de décision désigne une méthode basée sur l'utilisation d'un arbre de décision comme modèle prédictif.
Dans ces structures d'arbre, les feuilles représentent les valeurs de la variable-cible et les embranchements correspondent à des combinaisons de variables d'entrée qui mènent à ces valeurs.

Random Forest

L'algorithme des forêts d'arbres décisionnels effectue un apprentissage sur de multiples arbres de décision entraînés sur des sous-ensembles de données légèrement différents.

Nearest Neighbors

a méthode des k plus proches voisins consiste à prendre en compte (de façon identique) les k échantillons d'apprentissage dont l’entrée est la plus proche de la nouvelle entrée x, selon une distance à définir.