

Projet d’estimation des prix d'immobilier

Rapport final du traitement des données

GLO-7027 Analyse et traitement de données massives

Ali ASSAFIRI 111 054 128

Rhita OULIZ 111 082 917

06 avril 2019

**Table des matières**

[**1.** **Introduction 10 pages** 3](#_Toc4658580)

[**2.** **Méthodologie** 3](#_Toc4658581)

[**2.1.** **Description de l’algorithme** 3](#_Toc4658582)

[**2.2.** **Description du fonctionnement de la méthode** 3](#_Toc4658583)

[**2.3.** **Décisions de design et d’implémentation** 3](#_Toc4658584)

[**3.** **Expérimentation** 3](#_Toc4658585)

[3.1. **Description des tests** 3](#_Toc4658586)

[**3.2.** **Résultats obtenus et statistiques pertinents** 3](#_Toc4658587)

[**3.3.** **Description des résultats et discutions** 3](#_Toc4658588)

[**4.** **Discutions** 3](#_Toc4658589)

[**4.1.** **Attributs utilisés** 3](#_Toc4658590)

[**4.2.** **Attributs importants** 3](#_Toc4658591)

[**4.3.** **Comparaison avec les attributs importants du rapport 3** 3](#_Toc4658592)

[**4.4.** **Problèmes identifiés et solutions** 3](#_Toc4658593)

[**5.** **Étude comparative des deux méthodes** (2 points) 3](#_Toc4658594)

[**5.1.** **Comparaison des deux solutions réalisées** 3](#_Toc4658595)

[**5.2.** **Condition de performance de chacune des deux méthodes** 3](#_Toc4658596)

[**5.3.** **Différences entre les deux méthodes** 4](#_Toc4658597)

[**6.** **Conclusion** (1 point) 4](#_Toc4658598)

[**6.1.** **Rétrospective sur le projet** 4](#_Toc4658599)

[**6.2.** **Améliorations et recommandation** 4](#_Toc4658600)

Table des tableaux

**Aucune entrée de table d'illustration n'a été trouvée.**

Table des figures

**Aucune entrée de table d'illustration n'a été trouvée.**

# **Introduction 10 pages**

Le présent rapport décrit notre deuxième traitement de données dans le but d’estimation de la valeur immobilière des maisons de la ville d’Âmes aux États-Unis. En plus de l’analyse et la description de ce traitement et ses résultats, une comparaison avec le traitement effectué lors du rapport 3 sera à la cinquième partie de ce rapport. Pour conclure avec des recommandations améliorations et rétrospectives.

À cette étape de traitement, deux méthodes seront implémentées en plus de la methode Random Forest implémenté précédemment. Il s’agit de la méthode de GBR (Gradient Boosting Regressor Model) et sa version légère en termes de temps de calcul LGBM (Light Gradient Boosting Regressor Model).

Une étude d’attributs les plus important pour chaque méthode sera implémenter également afin de découvrir les combinaisons intéressantes.

# **Méthodologie**

### **Description de l’algorithme**

En se basant sur la revue de la littérature faite dans les étapes précédentes de ce projet, on continue nos analyses en testant un nouvel algorithme comparé aux forêts aléatoires (anglais : Random Forest). Comme proposé antérieurement, l’algorithme mis à l’épreuve cette fois-ci est : les arbres de régression optimisées (en anglais : Gradient Boosting). Cet algorithme supervisé peut être utilisé à la fois pour la régression et la classification.

Dans le cas présent, on utilise que la régression puisque nous cherchons une prédiction numérique en tant que résultat final. Les arbres de régression optimisées peuvent être vues en deux parties : la première qui est les arbres de régression qui se résume à être des modèles qui associent une réponse à leurs prédicteurs par des divisions binaires récursives. La deuxième partie est le booster, ce dernier est une méthode adaptative permettant de combiner de nombreux modèles simples pour améliorer les performances prédictives.

D’un point de vu plus pratique, les arbres de régression optimisée incorporent des avantages importants des méthodes basées sur l’arbre, la gestion de différents types de variables prédictives et l’adaptation des données manquantes.

Ils n'ont pas besoin de transformation de données ou d'élimination des valeurs aberrantes, ils peuvent s'adapter à des relations non linéaires complexes et gérer automatiquement les effets d'interaction entre les prédicteurs.

### **Description du fonctionnement de la méthode**

Lorsque on essaie de prédire la variable cible à l’aide d’une technique d’apprentissage automatique, les principales causes de différence entre les valeurs réelles et prédites sont le bruit, la variance et les biais. Ceci est autant applicable dans le cas de la méthode des arbres de régression optimisés.

En bref, cette technique utilise la logique dans laquelle : les prédicteurs du premier niveau tirent des leçons des erreurs des prédicteurs précédents. Par conséquent, les observations ont une probabilité inégale d'apparaître dans les modèles ultérieurs et celles avec l'erreur la plus élevée apparaissent le plus. À savoir, les observations ne sont donc pas choisies en fonction du processus d'amorçage, mais en fonction de l'erreur.

Ainsi, les prédicteurs peuvent être choisis parmi une gamme de modèles tels que des arbres de décision, des régresseurs, des classificateurs, etc. Les nouveaux prédicteurs tirant des leçons des erreurs commises par les prédicteurs précédents, il faut moins de temps et d’itérations pour se rapprocher des prévisions réelles. Mais il faut choisir les critères d'arrêt avec soin, sinon cela peut conduire à une sur-apprentissage des données d’entraînement.

À cet effet, l’algorithme des arbres de régression optimisés peut être résumé par, une première modélisation des données avec des modèles simples. Une détection d’erreurs sur les résultats du modèle est faite pour une optimisation d’erreur. Ces erreurs permettent de détecter les valeurs aberrantes à ajuster le modèle. Ensuite, une combinaison de tous les prédicteurs en attribuant des poids à chaque prédicteur.

### **Décisions de design et d’implémentation**

Afin d’améliorer notre apprentissage, une augmentation de données a été effectuer en donnant aux données manquantes des valeurs représentatif.

La figure ci-dessus représente le pourcentage de données manquantes. L’enrichissement des données a été selon la stratégie suivante :

A titre d’exemple, pour les

Vu que cet algorithme est déjà implanté par une libraire appelée « scikit-learn » en python, il suffit en faire une bonne utilisation après avoir préparé les données. Cela étant dit, une étape préliminaire est de préparer les données pour qu’elle soit admissible dans l’algorithme des forêts aléatoires. Sachant que, certains hyper paramètres de la régression peuvent être choisis tel que mentionné précédemment. Les deux plus importants paramètres sont le nombre d’estimateurs et le maximum de profondeur.

Quant au nombre d’estimateurs, ce paramètre s’occupe de spécifier le nombre d’arbres dans une forêt. La valeur par défaut est de 10 arbres par forêt. Ce chiffre entre autres impacte indirectement le nombre de solutions qui seront proposées par l’algorithme, ce nombre est inversement proportionnel au nombre de variables en entrée.

Comme son nom l’indique, le maximum de profondeur se résume à être un entier qui détermine la profondeur des arbres dans chaque nœud. Autrement dit, si la valeur est None, les nœuds sont développés jusqu'à ce que toutes les feuilles (sous-branches) soient pures ou que toutes les feuilles contiennent moins d'échantillons possibles ou selon un nombre prédéterminé.

C’est deux paramètres ne peuvent être choisis arbitrairement, ainsi, des tests sont faits en variant ces paramètres et en fixant les attributs en entré, puis tester la performance du modèle en observant les valeurs de scores obtenus après chaque entrainement.

# **Expérimentation**

### **Description des tests**

* + 1. **Transformation logarithmique du prix**

Dans le but d’évaluer la pertinence de l'ajustement de la [distribution](https://fr.wikipedia.org/wiki/Loi_de_probabilit%C3%A9) donnée d’entrainement, nous avons générer le diagramme quantile-quantile. Il s’est avérer que la distribution n’est pas alignée sur la première bissectrice. La figure 2 à l’annexe B le démontre.

Après l’exploration de différentes transformations, nous avons trouvé que la transformation logarithmique de la distribution des prix de vente donne un diagramme quantile-quantile aligné sur la première bissectrice. La figure 3 de l’annexe B illustre ce fait.

Ainsi, nous avons testé la prédiction des prix de vente d’immobiliers avec et sans transformation logarithmique et évalué la performance des deux algorithmes des arbres de régression optimisées : GBR (Gradient Boosting Regressor Model) et LGBM (Light Gradient Boosting Regressor Model).

* + 1. **Stratégie de recherche de l’architecture performante du modèle**

Les paramètres qui influencent grandement la performance de la méthode de régression par descente de gradient sont le taux d’apprentissage, la profondeur maximale des arbres, nombres d’estimateurs et le nombre minimum d’exemple par noeud. Nous avons donné à ces paramètres différentes valeurs représenté au tableau ci-dessous et testé la performance du model.

|  |  |  |  |
| --- | --- | --- | --- |
| Paramètres d’apprentissage | Valeur 1 | Valeur 2 | Valeur 3 |
| Taux d’apprentissage | 0.1 | 0.05 | 0.01 |
| Profondeur maximale | 2 | 4 | 6 |
| Nombre d’estimateurs | 100 | 500 | 1000 |
| Nombre min d’exemple par nœud | 3 | 9 | 18 |

Tableau 1: Les valeurs des paramètres du modèle à tester

Pour ce faire, la fonctionnalité de *GridSearchCV()* de librairie *Scikit-learn* nous a permis de trouver la combinaison des valeurs des paramètres la plus performante.

* + 1. **Réduction de dimensionnalité**

Dans le but de chercher plus de performance avec un cout de calcul plus optimal. Nous avons déterminé les attributs les plus importants pour les deux modèles. Et ce en utilisant les fonctionnalités de la librairie *Scikit-learn* toujours. Pour cette étape, nous avons appelé la fonction *transform()* après l’entrainement du modèle.

### **Résultats obtenus et statistiques pertinents**

* + 1. **Transformation logarithmique du prix**

Pour la méthode LGBM, nous avons constaté une amélioration des résultats avec les données transformées. Cependant, le modelé GBR le taux de précisions ont diminué par rapport aux résultats d’entrainement sans transformation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Statistiques d’apprentissage | GBR | GBR | LGBM | LGBM |
| Transformation logarithmique | sans | avec | sans | avec |
| Taux de précision au test | 99,41 % | 97,25 % | 99,94 % | 98,90 % |
| Taux de précision à la validation | 93,44 % | 91,38% | 92,60% | 88,07 % |
| Taux de précision sur toutes les données | 98,26 % | 95,99 % | 98,52 % | 96,95 % |
| Durée de compilation | 5.22s | 1.52s | 2.13s | 0.25s |

Tableau 2: Taux de précision avec et sans transformation logarithmique

Le meilleur taux de précision revient au modèle GBR entrainé avec des données sans transformation. Ainsi, nous allons focaliser notre étude sue le model GBR même si le temps d’entrainement est le double de du model lège.

* + 1. **Stratégie de recherche de l’architecture performante du modèle**

|  |  |
| --- | --- |
| Paramètres d’apprentissage | Valeur |
| Taux d’apprentissage | 0.1 |
| Profondeur maximale | 6 |
| Nombre d’estimateurs | 100 |
| Nombre min d’exemple par nœud | 3 |

* + 1. **Réduction de dimensionnalité**

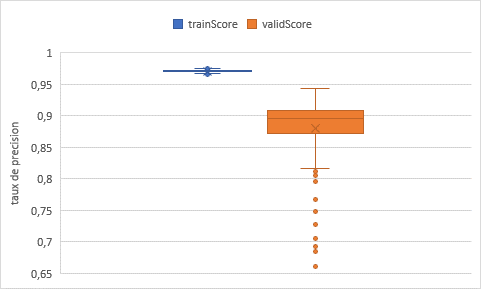


Figure 6: Taux de précision de GBR

### **Description des résultats et discutions**

Cas de maisons avec une mauvaise estimation

# **Discutions**

### **Attributs utilisés**

### **Attributs importants**

* + 1. Méthode GBR

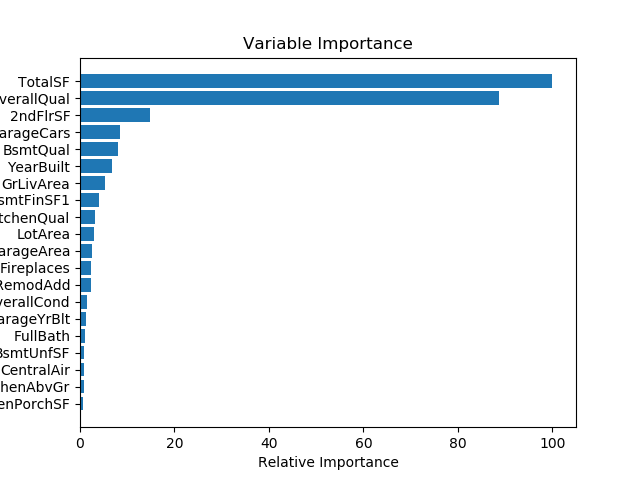


Figure 7: Gradient Boosting Regressor Model\_20importantFeatures

* + 1. Méthode LGBM

Voir Annexe B

Ajout d’un tableau plus clair

### **Comparaison avec les attributs importants du rapport 3**

Pour la méthode Randome forest, lors du rapport 3 on a travaillé avec les attributs …….

Une nouvelle étude sur la méthode nous a données. Voir l’annexe B

### **Problèmes identifiés et solutions**

# **Étude comparative des deux méthodes** (2 points)

### **Comparaison des deux solutions réalisées**

Selon

### **Condition de performance de chacune des deux méthodes**

Dans quelles conditions est-ce qu’une est préférable à l’autre, et pourquoi?

### **Différences entre les deux méthodes**

Discutez autant les différences provenant de la nature des algorithmes, que les différences résultant des leçons prises dans le rapport précédent.

# **Conclusion** (1 point)

### **Rétrospective sur le projet**

En quoi avez-vous eu raison dans votre plan initial, et en quoi avez-vous eu tort?

### **Améliorations et recommandation**

Si le projet était à refaire, que feriez-vous différemment?

**Annexe A : Graphes de visualisation des données**

Soumettez votre résultat final à Kaggle et indiquez votre score et position.

Incluez également la version finale de votre code (lien vers votre dépôt GIT ou fichier zip) avec votre soumission.

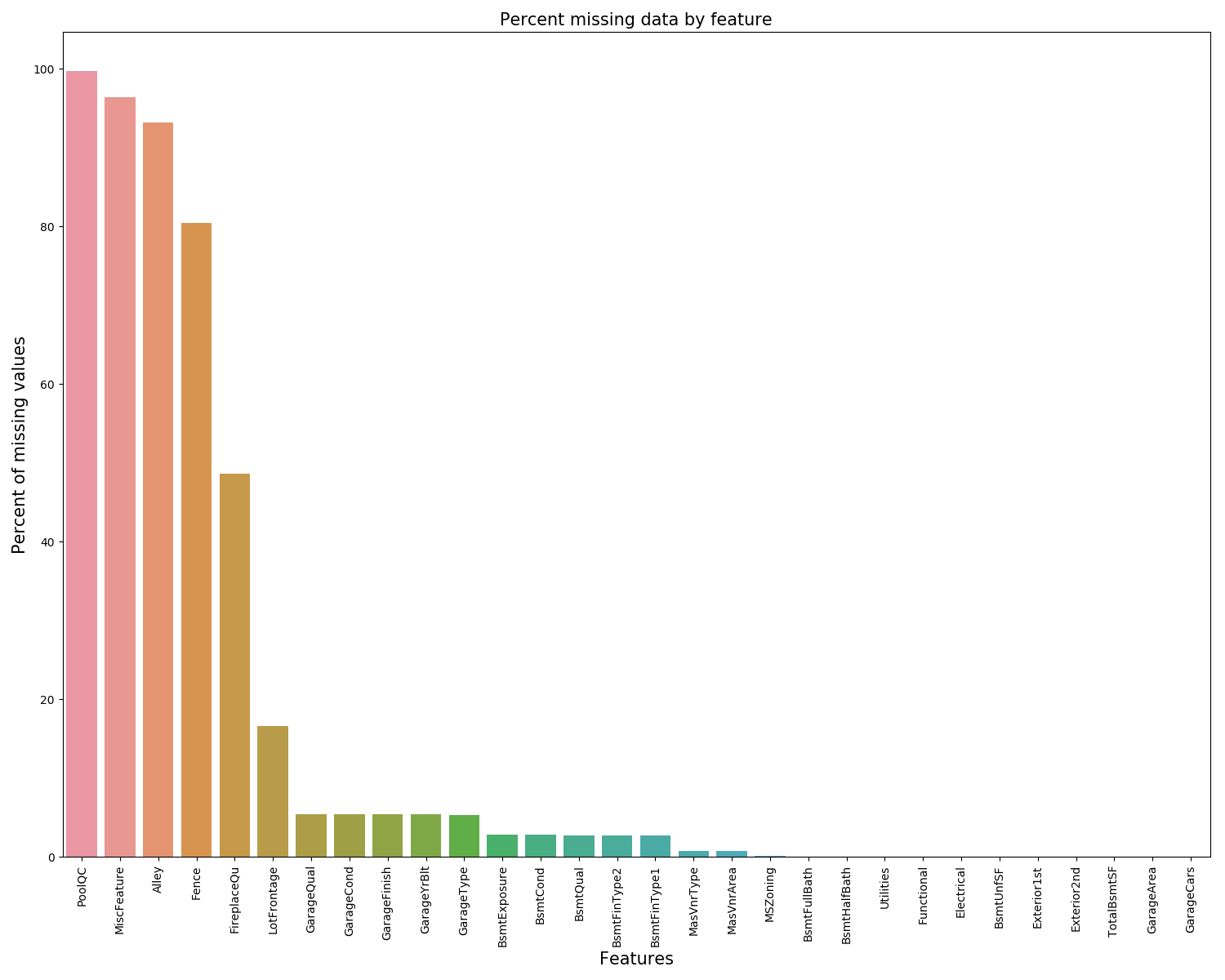


Figure 8: missingData

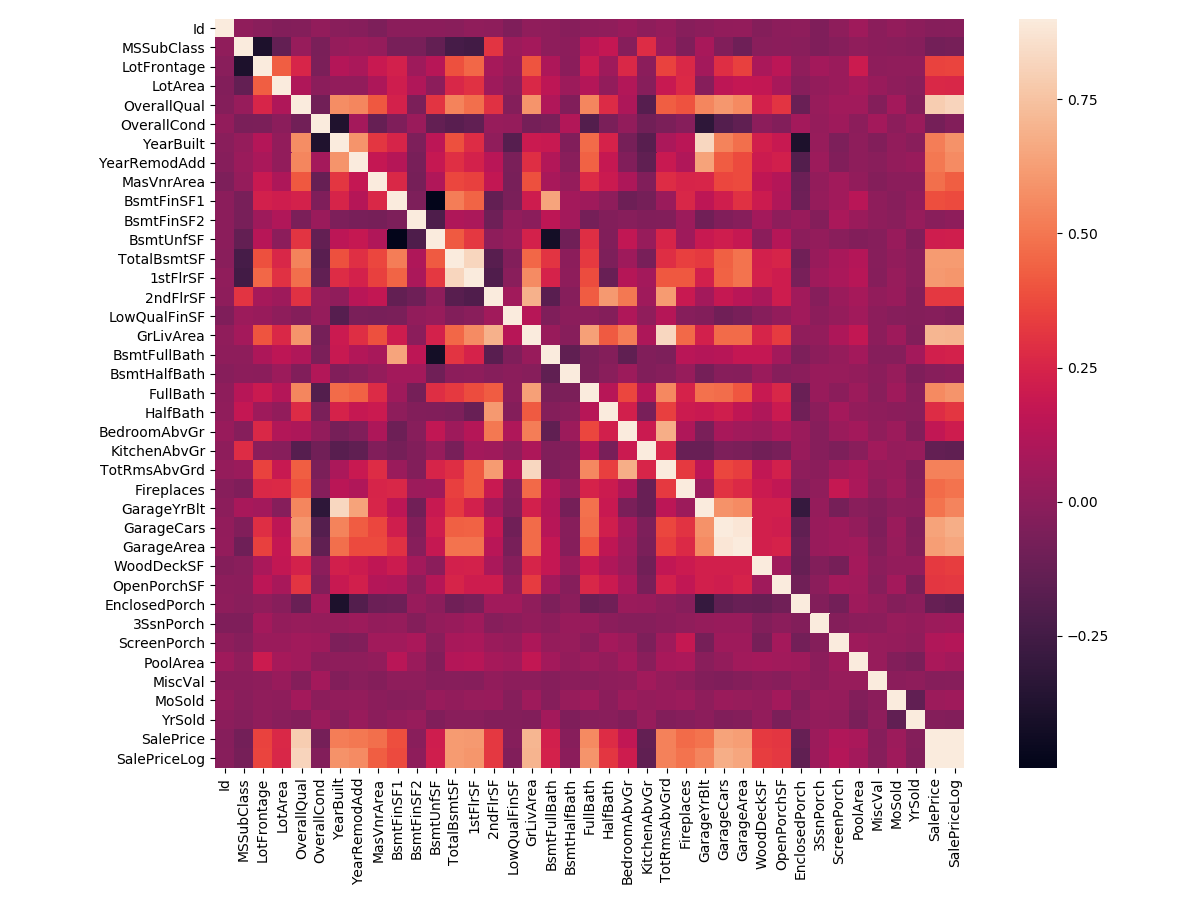


Figure 9: Corrélation entre les attributs après transformation des données

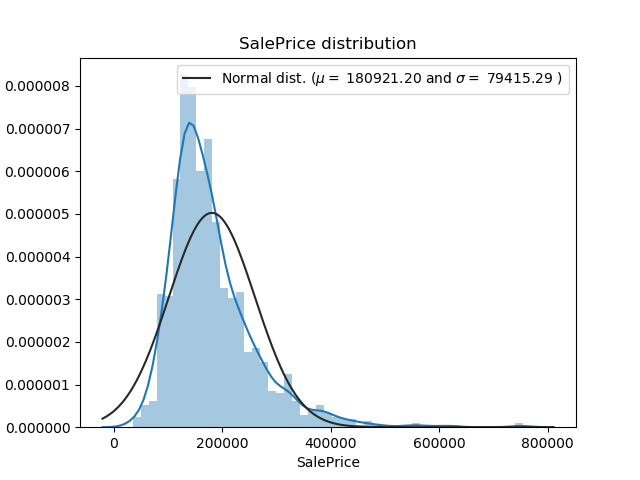


Figure 1: SalePrice\_distrubution sans transformation logarithmique

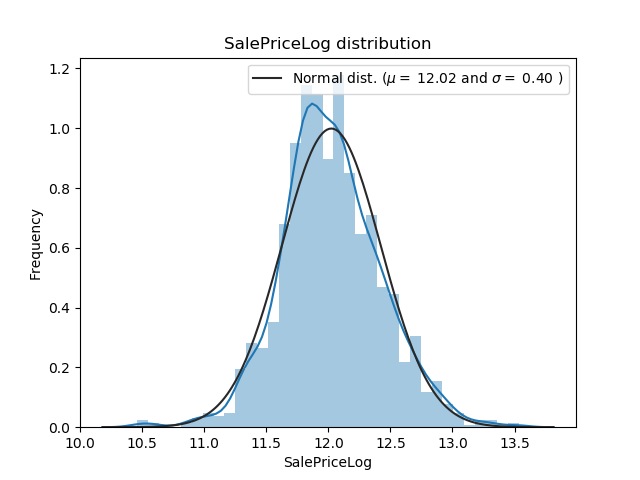


Figure 2: SalePriceLog\_distrubution

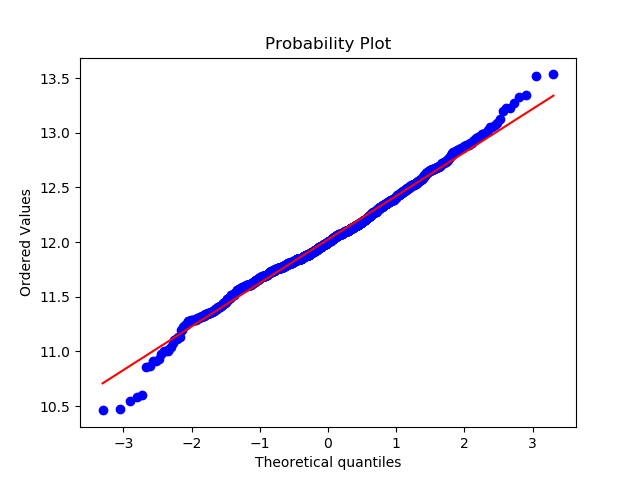


Figure 3: SalePriceLog\_QQ-plot

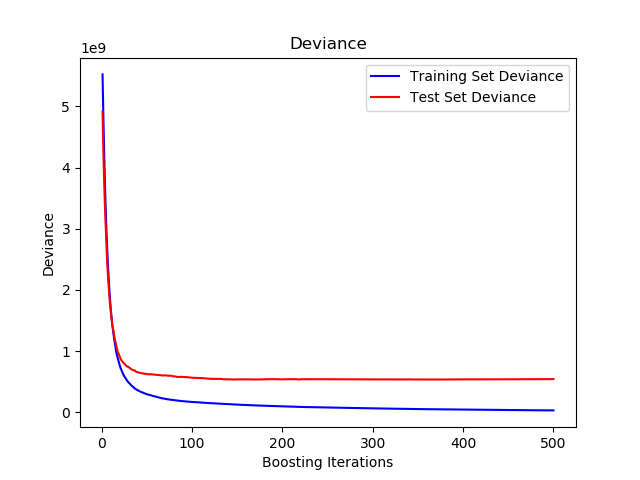


Figure 4: Gradient Boosting Regressor Model\_Deviance

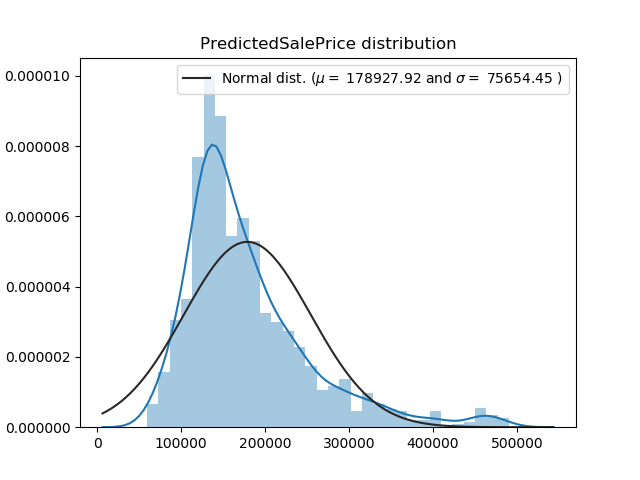
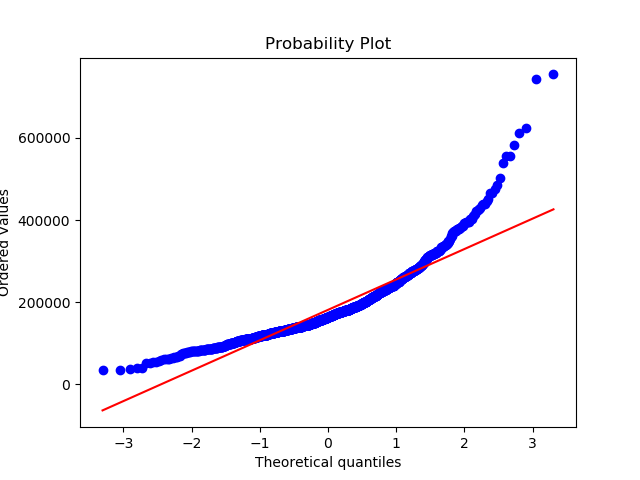


Figure 5: PredictedSalePrice\_distrubution

Figure 10: SalePrice\_QQ-plot

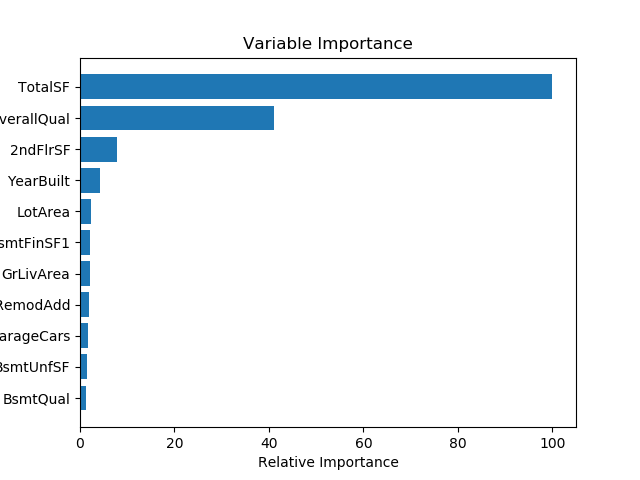


Figure 11: Random Forest Regressor Model\_10importantFeatures

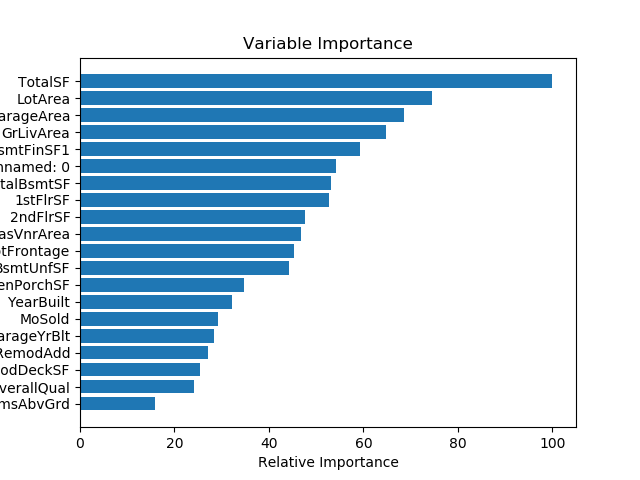


Figure 12: Light Gradient Boosting Regressor Model\_20importantFeatures

**Annexe B : Resultats des tests**

Feature selection from models

\*\*\*\*\*\* Random Forest Regressor Model \*\*\*\*\*\*\*\*\*

Train score: 0.9803

Validation score: 0.9051

All training data score: 0.9658

Random Forest Regressor Model time: 20.20s

New train\_X shape (1168, 19)

Feature selection from model

new features shape (1168, 2)

\*\*\*\*\*\* Random Forest Regressor Model White Importent Fetures \*\*\*\*\*\*\*\*\*

Train score: 0.9806

Validation score: 0.9037

Random Forest Regressor Model White Importent Fetures time: 8.12s

Feature selection from model

new features shape (1168, 17)

\*\*\*\*\*\* Gradient Boosting Regressor Model \*\*\*\*\*\*\*\*\*

Train score: 0.9941

Validation score: 0.9344

All training data score: 0.9826

Gradient Boosting Regressor Model time: 5.22s

New train\_X shape (1168, 17)

Feature selection from model

new features shape (1168, 2)

\*\*\*\*\*\* Gradient Boosting Regressor Model White Importent Fetures \*\*\*\*\*\*\*\*\*

Train score: 0.9913

Validation score: 0.9235

Gradient Boosting Regressor Model White Importent Fetures time: 0.97s

Feature selection from model

new features shape (1168, 39)

\*\*\*\*\*\* Light Gradient Boosting Regressor Model \*\*\*\*\*\*\*\*\*

Train score: 0.9994

Validation score: 0.9260

All training data score: 0.9852

Light Gradient Boosting Regressor Model time: 2.13s

New train\_X shape (1168, 39)

Feature selection from model

new features shape (1168, 17)

\*\*\*\*\*\* Light Gradient Boosting Regressor Model White Importent Fetures \*\*\*\*\*\*\*\*\*

Train score: 0.9995

Validation score: 0.9153

Light Gradient Boosting Regressor Model White Importent Fetures time: 1.43s

**Annexe C**

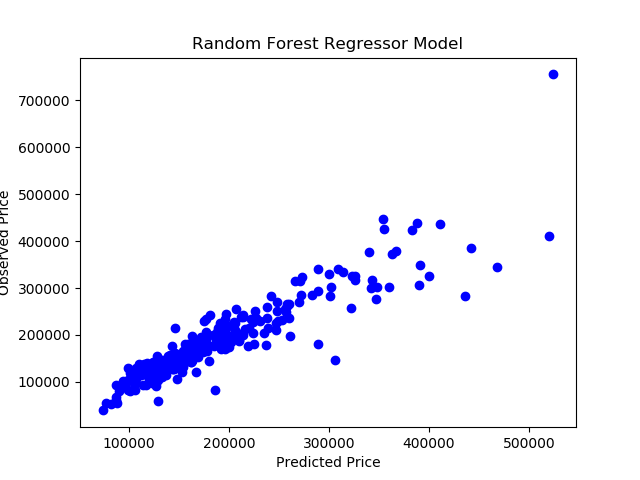
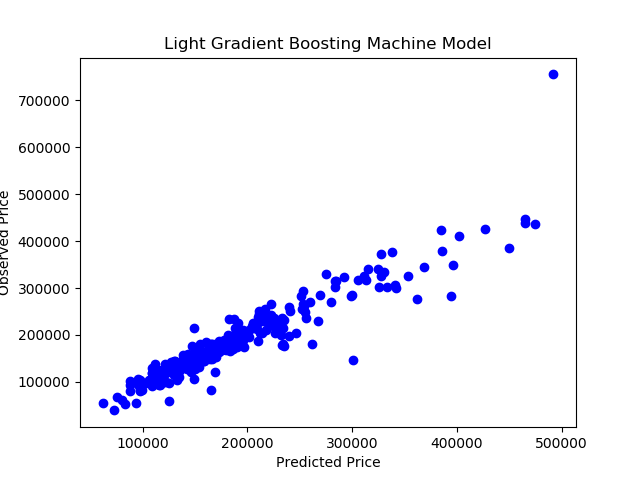


Figure 13: Random Forest Regressor Model\_distrubution

Figure 14: Light Gradient Boosting Machine Model\_distrubution

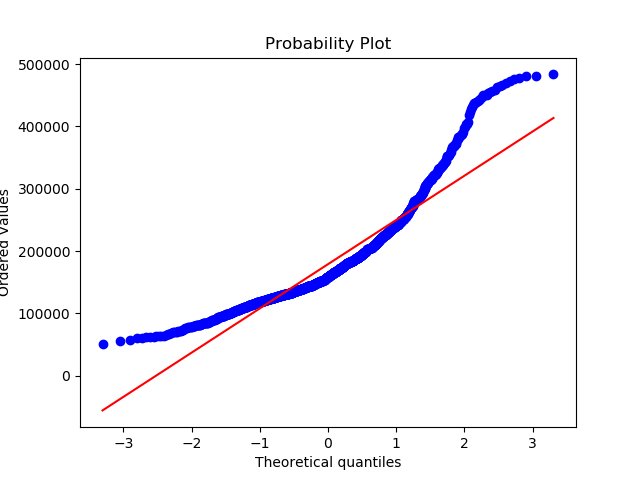


Figure 15: PredictedSalePrice\_QQ-plot