Advanced MachineLearning

TD 2 : Model optimization

Sujet : Optimisation d'un modèle de Machine Learning

Yanis FADILI & Lies HAOUAS
Professeur : Denis OBLIN

Sommaire

2. Sélection du modèle
3. Exploration du modèle
4. Optimisation du modèle

1. Introduction

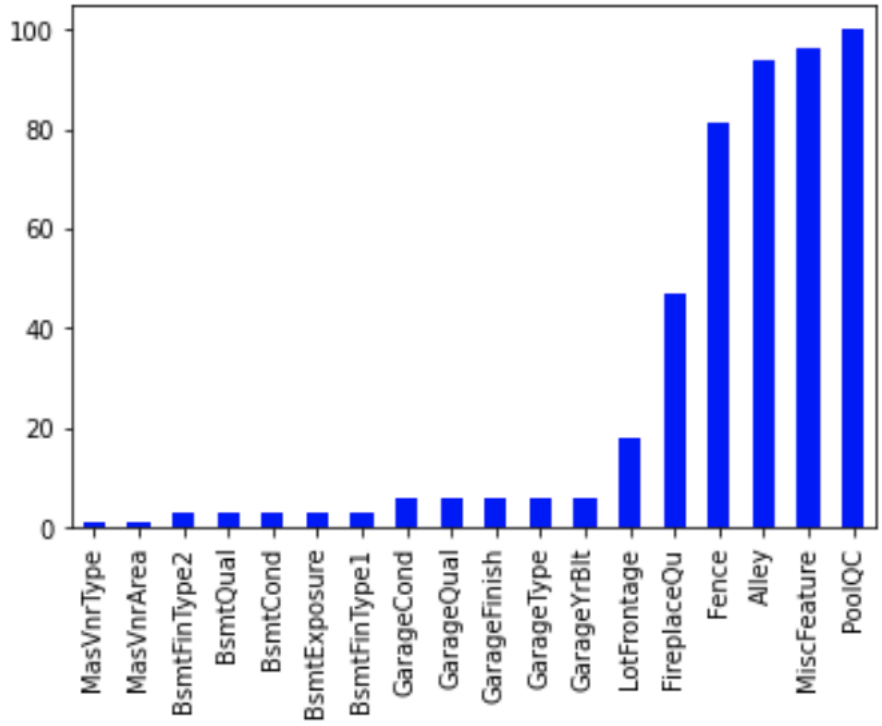
L'objectif du projet est ici de travailler sur un modèle de notre choix en se basant sur un jeu de données déjà optimal, puis de l'évaluer en vue de l'optimiser. On cherche ici à prédire la colonne SalesPrice.

Le projet est développé en Python 3 sur Google Colab, disponible sur le lien ci-dessous :

2. Data cleaning et exploration

Dans un premier, nous avons qu'il y avait 5 features avec plus de 50% de données manquantes, on a décidé de les supprimer car avec plus de 50% de valeurs manquantes, ceux

sont des features inexploitables en prenant compte le grand nombre de valeurs manquantes.



Ensuite, nous avons géré le cas des valeurs manquantes pour colonnes de type catégorie, en remplaçant les valeurs manquantes par "Missing", dans les colonnes de type catégorie, cela nous permet de ne pas supprimer ces colonnes car on n'a remarqué qu'il y avait une relation plus ou moins avec les valeurs à prédire. Ensuite, on a oneHotEncoding les colonnes de types catégories pour les permettre de les utiliser avec notre modèle.

Pour les colonnes de types dates contenant des années, On a voulu donner un peu plus de sens aux dates, car une année n'a pas assez de sens dans ce contexte et quand il sera utilisé par le modèle ce sera uniquement une valeur numérique. Pour lui donner du sens, on a calculé le nombre d'année qui s'est écoulé entre l'année de la vente et la construction de la maison ou du garage pour avoir une donnée plus parlante et plus exploitable.

Pour les colonnes de types numériques, nous avons géré le cas des valeurs manquantes en remplaçant les Nan par la médiane de la colonne. On a choisi la médianne et pas la moyenne car il y a beaucoup de outliers pour ces colonnes et pour ne pas avoir une valeur erronée par les outliers, on s'est dit que c'était un choix plus judicieux.

Pour le choix des features, on a choisi les 10 features les plus corrélées avec la valeur à prédire :

OverallQual	0.790982
GrLivArea	0.708624
Neighborhood	0.696882
ExterQual	0.682639
KitchenQual	0.659600
GarageCars	0.640409
GarageArea	0.623431
BsmtQual	0.622925
TotalBsmtSF	0.613581
1stFlrSF	0.605852

3. Les modèles

Nous avons utiliser 3 modèles de régression :
- La régression linéaire multiple
- XG Boost
- Random Forest

Le modèle de Random Forest est un modèle simple et basé sur l'idée d'arbres de décision qui va essayer de trouver les features les plus pertinentes en créant des multiples branches et ainsi choisir la plus intéressantes. C'est le modèle qui a la meilleure précision

Pour le random Forest, les paramètres que nous avons fait varier pour trouver les meilleurs sont le max_depth et le nombre minimal de feuilles:

```
randomForest = RandomForestRegressor()

param = {'n_estimators' : [int(x) for x in
np.linspace(start=80,stop=100, num=10)],
          'max_depth' : [90,100],
          'min_samples_split':[4,8],
          'min_samples_leaf':[2,4],
   }
```

```
gridSearch=GridSearchCV(randomForestAlgo,param,scoring='r2',cv=5)
gridSearch.fit(X_train_Scaled,y_train)
```

Voici le score obtenu avec les différents modèles :

Random Forest Regression	0.838931
Linear regression	0.811116
XG Boost regression	0.784357

Lien du collab :
https://colab.research.google.com/drive/16yciwE0Mv019zHVSilUlnq6ZAx_TZ_M?usp=sharing