TP N° 4 - Régression linéaire
Comment pouvons-nous utiliser un modèle de régression linéaire pour prédire les prix de vente de maisons ?

Contexte

Calculer le prix de vente d'une maison est souvent compliqué en raison du grand nombre de variables qui jouent sur les décisions de prix. Cependant, avec suffisamment de données, nous pouvons créer un modèle de *régression linéaire* qui peut aider à déterminer les variables (features) les plus importantes d'une maison et à prédire avec précision son prix de vente.

Ces informations peuvent être utilisées pour aider les propriétaires à maximiser la valeur de leur maison et à augmenter le prix de vente de leur bien sur le marché.

À l'aide d'un ensemble de données sur les logements d'Ames, dans l'Iowa (États-Unis), nous allons créer dans ce TP un modèle de régression linéaire qui prédit les prix de vente des logements, en nous basant sur plus de 2000 observations de logements et plus de 80 variables (features), comme la superficie en pieds carrés et la qualité de la cuisine etc ..., en utilisant diverses techniques d'ingénierie, de sélection et de régularisation des variables.

Dans ce TP, nous avons l'objectif suivant: créer un *modèle* à la fois *performant* et *généralisable*. Cela signifie que le modèle doit réagir à des ensembles de données inédits. Par la suite, nous cherchons à évaluer les performances de ce modèle en utilisant les métriques de régression vues en cours.

1 Analyse exploratoire de données

Télécharger le dossier **data** pour ce TP à partir de l'espace moodle (section 8). **data** contient 3 fichiers: `ames_prices_description.txt`, `ames_prices_train.csv` et `ames_prices_test.csv`. Le premier décrit les variables (features) du jeu données, tandis que les fichiers CSV sont les données d'apprentissage (train set) et du test (test set). Notre variable cible est la colonne `SalePrice` indiquant le prix d'une maison. Remarquer que cette colonne est vide pour les données test. Pour cela, notre but est d'ajuster un modèle de machine learning supervisé, en occurrence une régression linaire (voire pénalisée), en passant par les étapes d'analyse exploratoire des données (AED), les phases de nettoyage et pré-traitement des données (Pre-processing), la modélisation et finalement l'évaluation.

1.1 Importations des librairies et chargement des données

1. Importer toutes les librairies Python nécessaires pour le développement de vos modèles de prédiction.
2. Charger les données et les enregistrer dans une structure de données dataframe de Pandas.
3. Décrire succinctement les variables (on pourra lire et afficher le texte de la description à l'aide de la méthode `open`.
4. Quelles sont les dimensions de votre dataframe ?
5. Indiquer le type de chacune des variables.
6. Préciser la variable d'intérêt (variable dépendante) et les variables explicatives (variables indépendantes).
7. Visualiser la distribution de la variable cible. Interpréter.

1.2 Analyse univariée

Variable cilbe. Une première hypothèse de l'application d'une régression linéaire gaussienne est la normalité de la réponse Y=`SalePrice` sachant les features X = (`MSSubClass, MSZoning, ..., SaleType, SaleCondition`). Il s'agit donc de s'assurer que la variable continue Y= `SalePrice` est distribuée selon la loi normale. Un examen préalable des données à l'aide de graphique ou de statistiques simples peuvent déjà permettre de visualiser si la distribution empirique suit une loi normale.

1. Représenter la variable `SalePrice` à l'aide de l'histogramme de fréquence et regarder si elle semble s'ajuster à une distribution normale.
2. La loi normale est caractérisée par un coefficient d'asymétrie et un coefficient d'aplatissement nuls. Ces indicateurs peuvent donc également nous donner une idée. Calculer les coefficients d'asymétrie et d'aplatissement de la variable cible. Interpréter.
3. Le diagramme Quantile-Quantile" ou "diagramme Q-Q" ou "Q-Q plot" (`stats.probplot`) est un outil graphique permettant d'évaluer la pertinence de l'ajustement d'une distribution donnée à un modèle théorique. Si la série statistique suit bien la distribution théorique choisir (gaussienne), on devrait avoir les quantiles observés égaux aux quantiles associés au modèle théorique. Tracer le digramme Q-Q plot par rapport à la loi normale. Interpréter.

1.3 Analyse bivariée

1. Donner la matrice de corrélation pour les variables quantitatives. Interpréter.
2. Donner les corrélations entre la variable cible et les autres variables quantitatives. Interpréter.
3. Visualiser les distributions de certaines variables quantitatives (selon votre choix) et de la variable cible. Interpréter.
4. Visualiser les nuages de points de certaines variables catégorielles (selon votre choix) et de la variable cible. Interpréter.
5. Si la condition de normalité est légèrement violée, on pourrait faire une transformation logarithmique, $Y_{\text{transform}} = \log(Y + 1)$ pour la rendre approximativement normale.

2 Pré-traitement des données

1. On rappelle que le processus du pré-traitement de données se fait conjointement sur les données train et test. Il serait intéressant de concaténer ces deux données en un seul dataframe pour pouvoir appliquer en une seule fois les éventuelles stratégies de pré-traitement.
2. A l'aide de la fonction heatmap de Pandas, visualiser les valeurs manquantes des variables, si elles existent.
3. Calculez les pourcentages des valeurs manquantes de chaque variable. Il est préférable d'ordonner ces pourcentages.
4. Traiter les valeurs manquantes. Quelle stratégie adoptez-vous ?
5. Traiter les valeurs catégorielles. Quelle stratégie adoptez-vous ?
6. Utiliser une méthode pour normaliser les données quantitatives.

3 Modélisation + Évaluation

Après la phase de pré-traitement des données, nous allons procéder à la modélisation de la variable cible `SalePrice` à l'aide de la régression linéaire.

3.1 Régression linéaire

1. Entraîner un modèle de régression `linreg = LinearRegression()` sur les données d'apprentissage.
2. Évaluer le modèle sur les données d'apprentissage. Interpréter.
3. Évaluer le modèle sur les données de test. Interpréter.
4. Tracer l'histogramme des erreurs sur les données test.
5. Tracer la distribution des résidus $\hat{\varepsilon} = y - \hat{y}$. Interpréter.
6. Tracer les valeurs des coefficients $\boldsymbol{\theta}$ en utilisant `linreg.coef`. Interpréter.

3.2 Construction de pipelines d'apprentissage

Un pipeline (ou un transformateur) en machine learning peut être créé en rassemblant une séquence d'étapes impliquées dans la préparation d'un modèle. Il peut être utilisé pour automatiser le processus d'apprentissage. Le pipeline peut comprendre le prétraitement, la sélection des variables, la classification/régression. Les applications plus complexes peuvent avoir besoin d'intégrer d'autres étapes nécessaires dans ce pipeline.

Pour commencer, consulter la déclaration et le fonctionnement des pipelines dans l'API scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html

1. Construire deux pipelines qui traitent les variables qualitatives et quantitatives.
2. Construire un transformateur de colonnes (nouveau pipeline) qui regroupe les deux pipelines précédents. Utiliser le module `make_column_transformer` du `sklearn.compose` (voir un exemple détaillé ci-après: https://scikit-learn.org/stable/modules/generated/sklearn.compose.make_column_transformer.html.

3.3 Modèles prédictifs: régression linéaire pénalisée, SVR, KNN

La régression Ridge, Lasso et ElasticNet sont des extensions de la régression linéaire par moindres carrées permettant d'éviter le risque de sur-apprentissage. L'idée est d'ajouter une pénalisation au problème de régression par moindres carrées.

1. À l'aide des pipelines construits, entraîner 5 modèles de régression Ridge `linregRidge`, Lasso `linregLasso` et `linregElast`, `linregSVR` et `linregKNN` sur les données d'apprentissage. Penser à fixer le random-state pour la reproductibilité de vos résultats.
2. Évaluer les modèles sur les données d'apprentissage.
3. Évaluer les modèles sur les données de test.
4. Comparer les résultats avec ceux obtenus par la régression linéaire.
5. Optimiser les hyper-paramètres dans chaque modèle en utilisant la validation croisée sur une grille de valeurs (GridSearchCV).
6. Commentez les résultats obtenus. Quel algorithme recommanderiez-vous d'utiliser? ?

4 Allez plus loin: améliorer les performances de prédiction

Vous pouvez adopter autres stratégies de remplissage des valeurs manquantes et/ou travailler sur d'autres variables (features), ...