Projet Machine Learning : Les Véhicules d'occasion

Réalisé par :

DERRAZ Ouissal ,

ABBOU Touria,

OUADOUD Malika,

ASBAI Halima

M'HAJAR Chaymae

Sommaire :

1-Définition du machine learning:

Le Machine Learning est la science visant à créer des algorithmes capables d'apprendre des comportements, ou concepts, à partir d'exemples qu'ils observent ; un peu comme un enfant apprend de son environnement lorsqu'il grandit.

Voici ci-dessous quelques exemples récents d'applications directes du Machine Learning :

Siri : Il s'agit là de la reconnaissance de la parole, où l'algorithme s'adapte à l'accent et la façon de parler du locuteur, et améliore la précision de la compréhension au fil du temps.

Le thermostat Nest : Celui-ci apprend les préférences de température de son possesseur, et ce suivant les différents moments de la journée. Il baisse alors intelligemment la température lorsque personne n'est dans la maison pour ne pas gaspiller d'énergie.

La Google car : Il s'agit de la voiture autonome, soit sans conducteur humain, développée par Google. Celle-ci

2- L’idée du Projet :

Les prix des voitures neuves dans l'industrie sont fixés par le constructeur avec quelques coûts supplémentaires supportés par le gouvernement sous forme de taxes. Ainsi, les clients qui achètent une nouvelle voiture peuvent être assurés de l'argent qu'ils investissent pour en être dignes. Mais en raison de l'augmentation du prix des voitures neuves et de l'incapacité des clients à acheter de nouvelles voitures en raison du manque de fonds, les ventes de voitures d'occasion sont en augmentation mondiale (Pal, Arora et Palakurthy, 2018). Il existe un besoin pour un système de prédiction de prix de voiture d'occasion pour déterminer efficacement la valeur de la voiture en utilisant une variété de caractéristiques. Même s'il existe des sites Web qui offrent ce service, leur méthode de prédiction n'est peut-être pas la meilleure. En outre, différents modèles et systèmes peuvent contribuer à prédire la puissance pour la valeur marchande réelle d'une voiture d'occasion. Il est important de connaître leur valeur marchande réelle lors de l'achat et de la vente.

3-les outils :

a-dataset

Les données utilisées dans ce projet ont été téléchargées depuis Kaggle. elles sont été téléchargés sur Kaggle par Austin Reese, utilisateur de Kaggle.com. Austin Reese a récupéré ces données de craigslist à des fins non lucratives. Il contient la plupart des informations pertinentes fournies par Craigslist sur les ventes de voitures, y compris des colonnes telles que le prix, l'état, le fabricant, la latitude/longitude et 22 autres catégories.

b-pré-traitement

Vérification des valeurs nulles :

- La première étape du nettoyage des données consistait à étudier le nombre de points nuls et le pourcentage de points de données nuls. Mais aucune valeur nulle n'est détecté, alors on avait pas besoin de supprimer les valeurs nulles.

Suppression des colonnes inutiles

Dans une prochaine étape on a supprimé les colonnes inutiles. À cette fin, les colonnes « Car_Name », «Current Year», « Year » ont été totalement supprimées.

Etudier la relation entre les colonnes: -

Comme deuxième étape, on a utilisé « describe() » , La fonction a renvoyé le résumé des statistiques de la base de données. Nous n'avons passé aucun paramètre, donc, la fonction a utilisé toutes les valeurs par défaut.

Le but c'est d'avoir une idée générale sur l'effet de chaque colonne sur les autres.

Encodage binaire :

Lorsque l'on est en présence de données **catégorielles textuelles**, il est nécessaire d'encoder nos variables. On parle d'**encodage binaire** lorsque l'on encode une variable en deux catégories, en général en 0, 1.

La stratégie habituellement adoptée est néanmoins celle de la création de **dummies**. A partir d'une variable, pour les n-catégories qui la compose, on créera n-1 colonnes encodées de façon binaire (0-1) correspondant, chacune correspondant à une des catégories comprises dans la variable. On retire une colonne car cette colonne peut être déduite des autres. Il est important que les variables ne soient pas corrélées pour la qualité de la prédiction.

c-La visualisation des données

La visualisation de données est la présentation de données sous forme d'images. C'est extrêmement important pour l'analyse des données, principalement en raison de l'écosystème fantastique de packages Python centrés sur les données. Et cela aide à comprendre les données, aussi complexes soient-elles, leur importance en résumant et en présentant une énorme quantité de données dans un format simple et facile à comprendre et aide à communiquer les informations de manière claire et efficace.

Pour cela on a choisi d'utiliser **pairplot(final_dataset)**.

d- Mettre en œuvre plusieurs algorithmes d'apprentissage pour la résolution de la problématique

target : Selling price

features : d'autre colonnes

```
X=final_dataset.iloc[:,1:]
y=final_dataset.iloc[:,0]#selling price : target
```

Les algorithme qu'on a utilisé :

***1-k plus proche voisin* :**

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import GridSearchCV
import numpy as np
```

```
param_grid = {'n_neighbors': np.arange(1, 20),
              'metric': ['euclidean', 'manhattan']}

grid = GridSearchCV(KNeighborsRegressor(), param_grid, cv=5)

grid.fit(X_train, y_train)
```

Score du testSet : 0.01

2-regression linéaire à plusieurs variables :

```
from sklearn import linear_model
reg = linear_model.LinearRegression()
reg.fit(X_train,y_train)
```

Score du testSet : 0.85

2-regression linéaire à plusieurs *variables :*

```
from sklearn import linear_model
reg = linear_model.LinearRegression()
reg.fit(X_train,y_train)
```

Score du testSet : 0.85

3- Bayesian ridge régression:

```
from sklearn import linear_model
reg1 = linear_model.BayesianRidge()
reg1.fit(X_train, y_train)
```

Score du testSet: 0.85

4- arbre de décision :

```
from sklearn import tree
model = tree.DecisionTreeRegressor()
model.fit(X_train, y_train)
model.score(X_test, y_test)
```

Score de testSet : 0.91

5-random forest :

```
rf = RandomForestRegressor(n_estimators= 1100, min_samples_split= 2,min_samples_leaf= 1, max_features= 'auto',
 max_depth= 20)
```

```
rf.fit(X_train,y_train)
```

Score de testSet : 0.91

6- stacking : est un algorithme qui permet d'assembler plusieurs algorithmes, en choisissant en estimateur final.

```
#stacking en combinant random forest et arbre de décision
from sklearn.ensemble import StackingRegressor
model = StackingRegressor([('regression a pls vars', rf),
                           ('arbre de decision', model)],
                           final_estimator=reg)

model.fit(X_train, y_train)
model.score(X_test, y_test)# 0.92 comme score
```

Score de testSet : 0.92

e-justification du choix d'algorithme :

le target est le prix de vente d'une voiture, on constate que ce prix change constamment, ce qui explique qu'on doit utiliser la régression, de ce fait on a fait 6 algorithme de régression, qui sont mentionnés ci-dessus.

on a choisi de travailler avec le modèle StackingRegressor, car c'est celui qui a le score le plus élevé : 0.92

f- grille d'évaluation des algorithmes avec les différentes métriques :

Pour l'algorithme random forest : on a utiliser RandomizedSearchCV pour déterminer les meilleurs hyper paramètres, on n'a pas choisi de travailler avec gridSearchCV, car on a 100 combinaisons, ce qui nécessite plus de temps.

```
andom = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,scoring='neg_mean_squared_error', n_iter = 10,
```

Et pour le modèle k plus proche voisin, on a utilisé gridSearchCV pour préciser les meilleurs hyper paramètres à utiliser.

```
param_grid = {'n_neighbors': np.arange(1, 20),
              'metric': ['euclidean', 'manhattan']}

grid = GridSearchCV(KNeighborsRegressor(), param_grid, cv=5)

grid.fit(X_train, y_train)
```

f- Conclusion :

le modèle stacking c'est celui qui a donné le plus grand score 0.92, dans ce modèle on a combiné random forest et l'arbre de décision et la regression avec plusieurs variables comme final estimateur.