Scikit Learn

Régression sur le prix des maisons à boston

1) Import de dataset

]:
```
1 from sklearn import datasets
```

2]:
```
1 boston = datasets.load_boston()
2 print(boston["DESCR"])
```

```
.. _boston_dataset:

Boston house prices dataset
---------------------------
```

1.1) récupération des X et Y

:
```
1 X = boston["data"]
2 y = boston["target"]
```

2) separer le jeu d'apprentissage du jeu de test

:
```
1 from sklearn.model_selection import train_test_split
```

:
```
1 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
```

3) choisir un algo et le tester

Documentation du Random Forest

:
```
1 from sklearn.ensemble import RandomForestRegressor
```

]:
```
1 algo        = RandomForestRegressor()
2 modele      = algo.fit(X_train, y_train)
3 performance = modele.score(X_test, y_test)
```

]:
```
1 performance
```

```
0.8597588038373313
```

4) grille de recherche

Documentation du Random Forest

Documentation grid search

Documentation grid search

]:
```
1 from sklearn.model_selection import GridSearchCV
```

]:
```
1 hyperparameters = {"max_depth" : [1,2,3]}
```

]:
```
1 grille = GridSearchCV(algo, hyperparameters)
```

]:
```
1 grille = grille.fit(X_train, y_train)
```

4.2) résultats

]:
```
1 meilleur_model = grille.best_estimator_
```

]:
```
1 grille.score(X_test, y_test)
```

```
0.8105203810236372
```

4.3) récupérer le modèle pour faire des prédictions

]:
```
1 une_ligne  = X_test[1]
2 prediction = meilleur_model.predict([une_ligne])
```

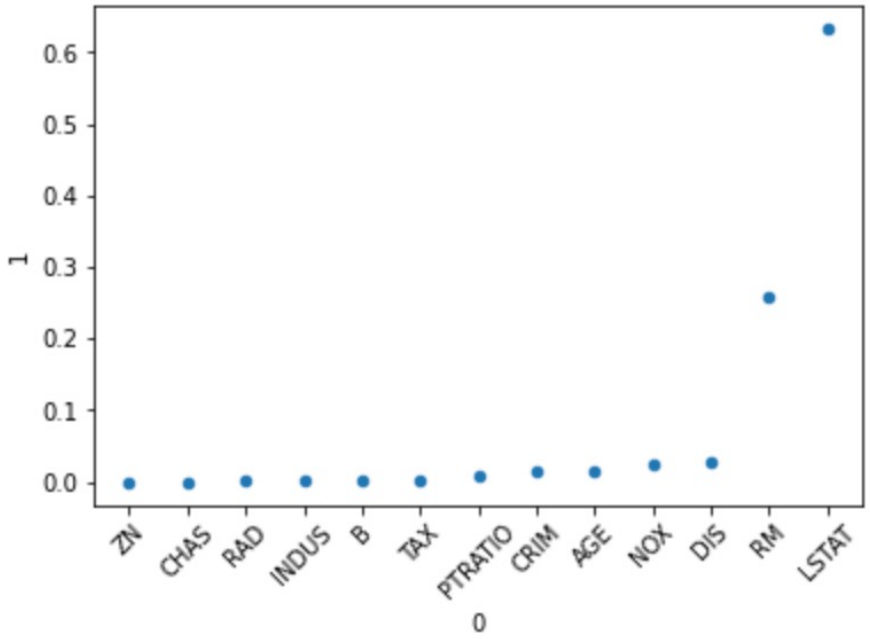
]:
```
1 prediction
```

```
array([22.13838832])
```

5) Features importances

```
|:  1 import pandas as pd
    2 from matplotlib import pyplot as plt
```

```
|:  1 feature_importances = list(zip(boston["feature_names"], meilleur_model.feature_importances_))
    2 feature_importances = pd.DataFrame(feature_importances)
    3
    4 feature_importances.sort_values(1).plot.scatter(x=0, y=1)
    5 plt.xticks(rotation=45) ;
```



5.1) on récupère de l'info de la documentation

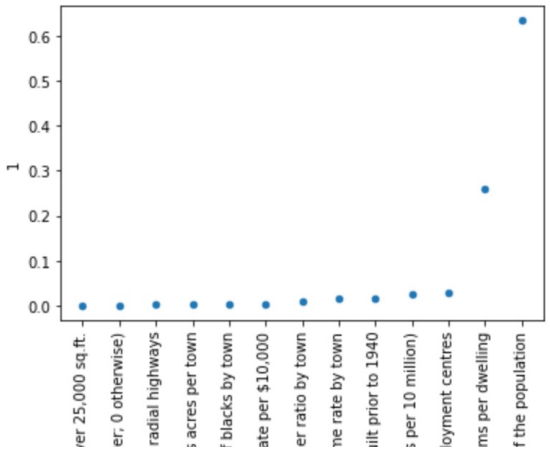
```
]:
signification_des_colonnes = {
    "CRIM"     : "per capita crime rate by town",
    "ZN"       : "proportion of residential land zoned for lots over 25,000 sq.ft.",
    "INDUS"    : "proportion of non-retail business acres per town",
    "CHAS"     : "Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)",
    "NOX"      : "nitric oxides concentration (parts per 10 million)",
    "RM"       : "average number of rooms per dwelling",
    "AGE"      : "proportion of owner-occupied units built prior to 1940",
    "DIS"      : "weighted distances to five Boston employment centres",
    "RAD"      : "index of accessibility to radial highways",
    "TAX"      : "full-value property-tax rate per $10,000",
    "PTRATIO"  : "pupil-teacher ratio by town",
    "B"        : "1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town",
    "LSTAT"    : "% lower status of the population",
    "MEDV"     : "Median value of owner-occupied homes in $1000's"}

```

On map le dictionnaire sur les features, pour créer une colonne de signification :

```
]:
feature_importances["signification"] = feature_importances[0].map(signification_des_colonnes)
```

on retrace le graph, avec en x la signification, et une rotation de 90

```
]:
feature_importances.sort_values(1).plot.scatter(x="signification", y=1)
plt.xticks(rotation=90) ;
```




```
signification_des_colonnes = {
   "CRIM"     : "per capita crime rate by town",
   "ZN"       : "proportion of residential land zoned for lots over 25,000 sq.ft.",
   "INDUS"    : "proportion of non-retail business acres per town",
   "CHAS"     : "Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)",
   "NOX"      : "nitric oxides concentration (parts per 10 million)",
   "RM"       : "average number of rooms per dwelling",
   "AGE"      : "proportion of owner-occupied units built prior to 1940",
   "DIS"      : "weighted distances to five Boston employment centres",
   "RAD"      : "index of accessibility to radial highways",
   "TAX"      : "full-value property-tax rate per $10,000",
   "PTRATIO"  : "pupil-teacher ratio by town",
   "B"        : "1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town",
   "LSTAT"    : "% lower status of the population",
   "MEDV"     : "Median value of owner-occupied homes in $1000's"}
```

Texte copiable (j'espère..)