Business & Decision

Formation

Machine Learning avec Python : Cours 3

08/11/2023

IAE NANTES
ÉCONOMIE & MANAGEMENT

VOTRE FORMATEUR

Cédric DANGEARD

Consultant Data

cedric.dangeard@businessdecision.com

Business & Decision

MACHINE LEARNING

OPTIMISATION

GRID SEARCH CV

GRID SEARCH

GridSearchCV est une fonction de sklearn.model_selection qui permet de tester différents jeux de paramètres pour un modèle donné et de comparer les performances.

GridSearchCV va tester toutes les combinaisons d'hyperparamètres.

Peut être très long, le choix des options doit prendre en compte le temps de calcul

RANDOMIZED SEARCH

Pour pallier le temps de calcul d'un GridSearch, une seconde approche est possible : Un randomized search (RandomizedSearchCV).

La fonction va essayer plusieurs combinaisons aléatoires d'hyperparamètres.

Requiert moins de temps de calcul qu'un GridSearch

Pas de garantie sur la qualité du modèle, ni de trouver la meilleure combinaison d'hyperparamètres.

GRID & RANDOM SEARCH

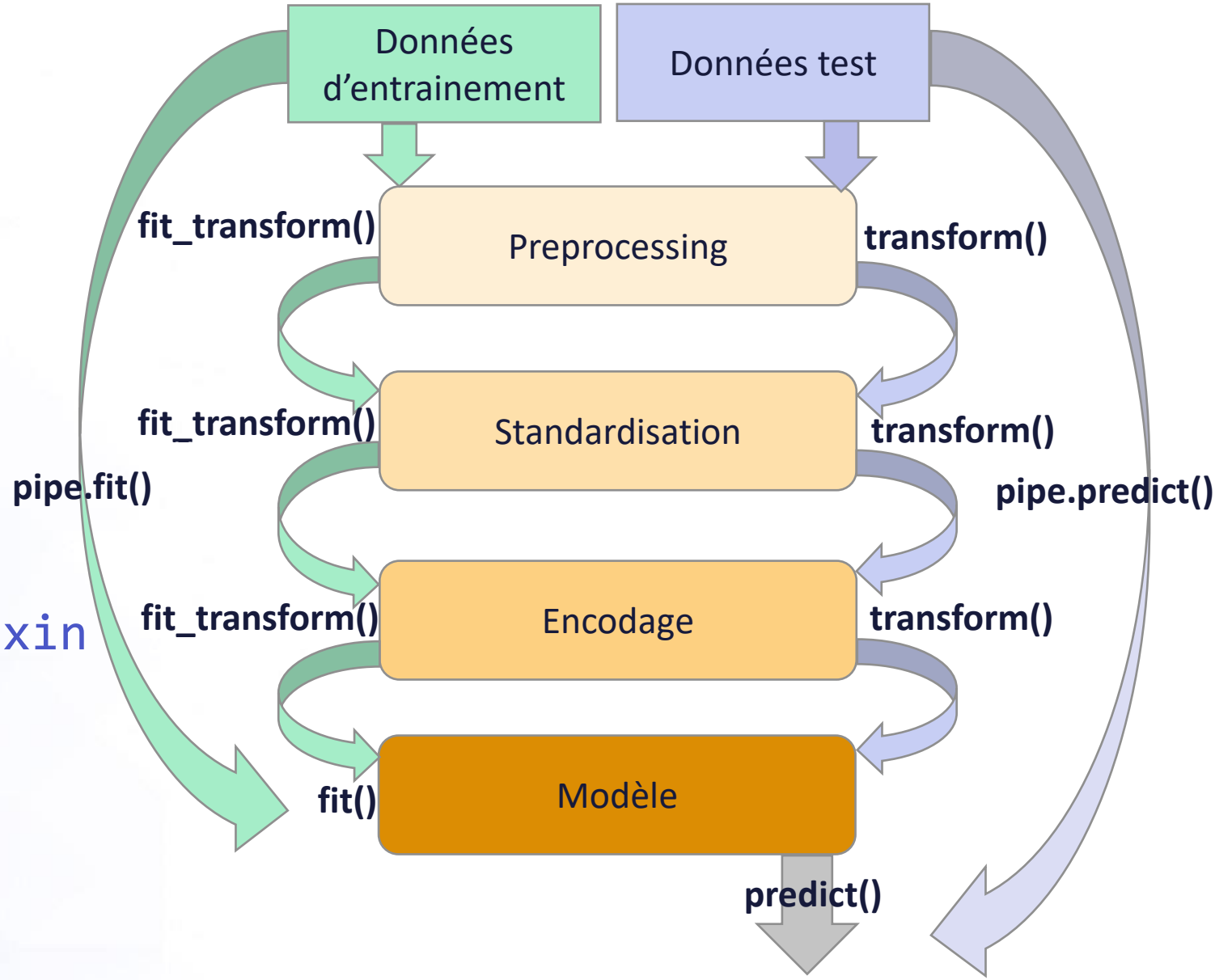
Paramètres :

- estimator : Le modèle à optimiser.
- param_grid: Dictionnaire de paramètres.
  - clef : nom du paramètre
  - value : liste de valeurs à tester
- scoring : Mesure de performance du modèle
- cv : On peut spécifier une stratégie de CV, par default utilise un 5-fold.
- refit : réentrainement du modèle sur la totalité du dataset.
- n_iter : Pour randomsearch, nombre de combinaisons d'hyperparamètres à tester.

```python
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
model = DecisionTreeClassifier()

param_grid = {
    'max_depth': [4, 6, 8, 10],
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'min_samples_split': [2, 3, 4, 5, 6, 7, 8, 9, 10]
}

grid_search = GridSearchCV(
    estimator=model,
    param_grid=param_grid,
    cv=3,
    n_jobs=-1
    )

grid_search.fit(Xtrain, ytrain)
print(grid_search.best_params_)
print(grid_search.best_score_)
```

PIPELINES

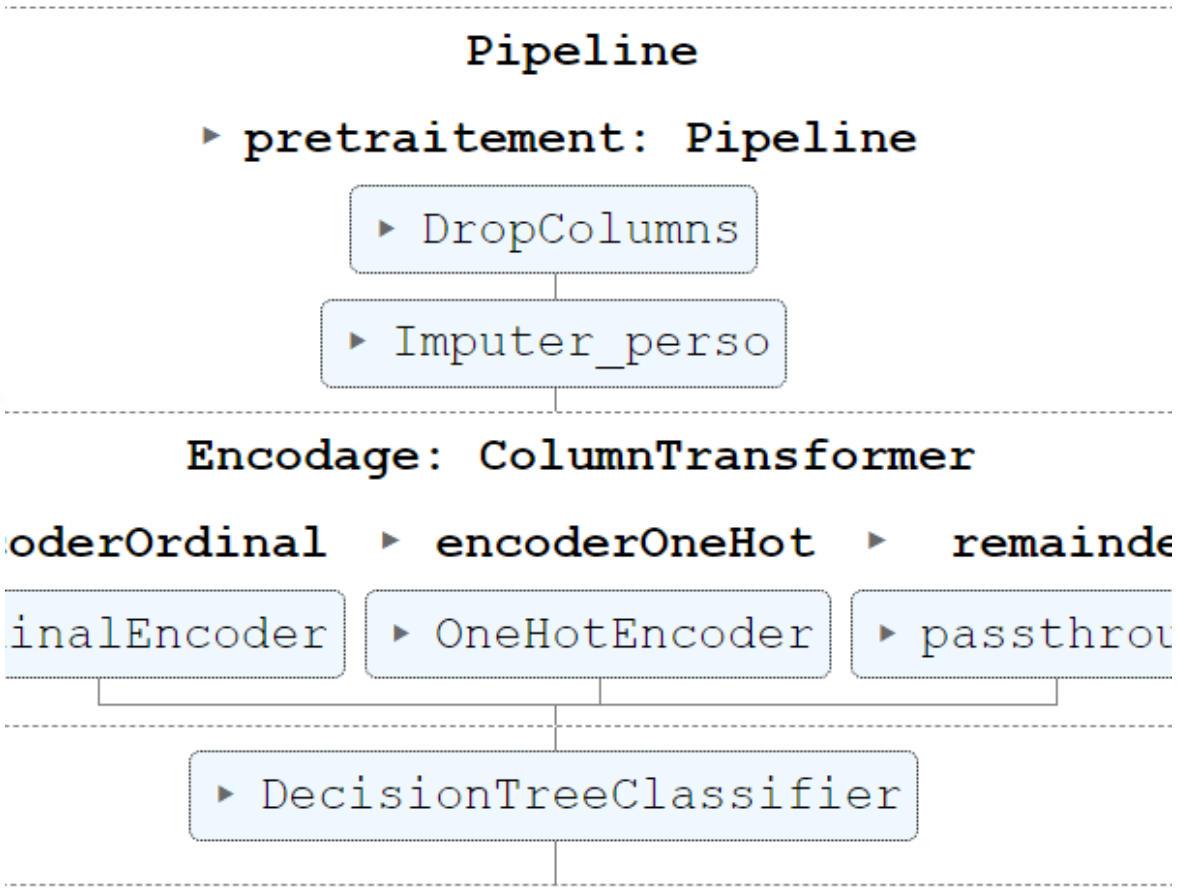
- Objectifs :
  - Automatiser les étapes de préparation des données et la modélisation
- `sklearn.pipeline.Pipeline`
  - Peut contenir un pipeline, un estimateur ou un transformer.
- `sklearn.base.TransformerMixin`
  - Pour créer ses propres transformations
- `sklearn.base.BaseEstimator`
  - Pour créer ses propres modèles



EXEMPLE PIPELINE

EXEMPLE PIPELINE

```
pretraitement = Pipeline([
    ('drop_columns', DropColumns(columnsToDrop = columns_to_drop)),
    ('imputer', Imputer_perso())
])

encodage = ColumnTransformer([
    ('encoderOrdinal',
        OrdinalEncoder(categories=categories),
        ordinalColumns),
    ('encoderOneHot',
        OneHotEncoder(drop='first', sparse=False),
        nominalColumns)
], remainder='passthrough'
)

pipeline = Pipeline([
    ('pretraitement', pretraitement),
    ('Encodage', encodage),
    ('model', DecisionTreeClassifier())
])
```



TP3 – OPTIMISATION

PROJET FINAL

PROJET FINAL

- Le projet vise à faire une analyse et une prédiction sur le jeu de donnés
- Lien des données (dispo sur kaggle aussi)

Pour le 13 février 2024

- Rendu sous forme de notebook
  - Avec votre d'équipe
  - Votre pseudo kaggle (si besoin)
  - A m'envoyer par mail : cedric.dangeard@businessdecision.com

- Bonus : participer sur Kaggle
  - Lien du Kaggle
  - 2 soumissions finales doivent être annotés dans le notebook.
  - Règles :
    - 20 soumissions par jour
    - 1 classement public
    - 1 classement privé

    (révélé à la fin)
  - Points bonus (en fonction de vos soumissions sur le classement privé)
    - 1 si vous passez la Baseline
    - 1 pour la première équipe

CONSIGNE PROJET

- Doit contenir :
  - Import du fichier
  - Une partie d'analyse des données
    - Chaque graphique doit être accompagné d'une courte explication de son intérêt
  - Une partie préparation des données
    - Bonus : sous forme de pipeline
  - Une partie modélisation dont
    - Un arbre
    - Un algorithme de bagging
    - Un algorithme de boosting
    - Une explication des paramètres choisis
  - Une partie optimisation des hyperparamètres

- Les critères de notation :
- Réalisation de chaque étape
- Les explications de chaque étape (le but est de voir que vous comprenez ce que vous exécutez : pourquoi on le fait ? explication des paramètres, quels scores utilisés pour performance du modèle ? quelle validation croisée ? comparaison des différents scores etc …)
- Rédaction du notebook (propreté du notebook, graphiques tous avec titres, légendes, code commenté si besoin, etc …)
- La recherche (n'hésitez pas à laisser des algos qui sont moins performants, mais expliquez pourquoi ils ne sont pas ceux que vous retiendrez)
- La performance de votre algo final et son interprétation (variables explicatives les plus importantes du modèle, score etc …)

MERCI POUR VOTRE ATTENTION

QUESTIONS ?