TP · GUIDE

TP Vin · un sommelier sans dégustation

Objectif et données

- Question : quelle note quality prédire pour un vin blanc ?
- 4 898 lignes ; 11 mesures physico-chimiques numériques.

Source : [S8] · CSV local

1. Charger et décrire

```
import pandas as pd
df = pd.read_csv('data/winequality-white.csv',
                 sep=';')
print(df.shape)
print(df['quality'].value_counts().sort_index())
```

- Ouvrir vin_jour1_exercices.ipynb ; examiner notes et manquants.
- Librairies : pandas, matplotlib, scikit-learn.

Source : [S8]

2. Réserver le test

```
from sklearn.model_selection import train_test_split
X = df.drop(columns='quality')
y = df['quality']
X_dev, X_test, y_dev, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
```

- Les variables sont numériques, mais pas à la même échelle.
- Ne consulter le test qu'après le choix final.

Source : [S2]

3. Baseline et erreur de métrique

```
from sklearn.dummy import DummyRegressor
from sklearn.metrics import mean_absolute_error
baseline = DummyRegressor(strategy='mean')
# Comparer ensuite un arbre libre
```

- Utiliser la MAE : erreur moyenne en points de note.
- Comparer MAE train et MAE validation pour voir la mémorisation.

4. Comparer sur les mêmes plis

```
from sklearn.model_selection import KFold
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Ridge
cv = KFold(5, shuffle=True, random_state=42)
ridge = make_pipeline(StandardScaler(),
                      Ridge(alpha=10))
```

- Ajouter moyenne, k-NN et arbre ; rapporter MAE et R^2.
- StandardScaler appartient au Pipeline, jamais au CSV entier.

Source : [S2] [S3]

5. Régulariser

```
from sklearn.model_selection import GridSearchCV
recherche = GridSearchCV(
    ridge, {'ridge__alpha': [0.1, 1, 10, 100, 1000]},
    cv=cv, scoring='neg_mean_absolute_error')
recherche.fit(X_dev, y_dev)
```

- Grand alpha = pénalisation Ridge plus forte.
- Comparer aussi plusieurs profondeurs d'arbre.

7. Test final et erreurs

```
modele_final = recherche.best_estimator_
predictions = modele_final.predict(X_test)
print(mean_absolute_error(y_test, predictions))
# Comparer à la baseline et examiner les erreurs
```

- Comparer coût, stabilité et performance
- Regarder notamment les notes rares et atypiques.

8. Sauvegarder et recharger

```
from joblib import dump, load
dump(modele_final, 'vin.joblib')
modele_recharge = load('vin.joblib')
# Vérifier la stabilité des prédictions
```

- Sauvegarder le Pipeline entier, normalisation comprise.
- Le notebook emploie un dossier temporaire pour ne pas laisser de fichier.

Références vérifiables

[S8] UCI · Wine Quality

https://archive.ics.uci.edu/dataset/186/wine+quality

[S2] scikit-learn · Pièges

https://scikit-learn.org/stable/common_pitfalls.html

[S3] scikit-learn · Validation croisée

https://scikit-learn.org/stable/modules/cross_validation.html