Classification - Bayésien Naïf

Classification Bayésienne Naïve

Théorie

Le classificateur **Bayésien Naïf** repose sur le **théorème de Bayes** et l'hypothèse d'indépendance conditionnelle entre les variables explicatives. Il est particulièrement efficace pour les problèmes de classification textuelle et fonctionne bien même avec peu de données d'entraînement.

Hyperparamètres

Contrairement à d'autres modèles, le **Bayésien Naïf** possède peu d'hyperparamètres. Cependant, le paramètre `var_smoothing` dans `GaussianNB` permet d'éviter les divisions par zéro en ajoutant un lissage aux variances estimées.

Nous allons rechercher la meilleure valeur de `var_smoothing` en testant plusieurs options.

Évaluation des performances

Lorsqu'on évalue un modèle de classification, plusieurs métriques sont utilisées :

- **Matrice de confusion** : Tableau qui résume les performances du modèle en comparant les vraies classes aux classes prédites. Les lignes correspondent aux **classes réelles**, et les colonnes aux **classes prédites**.
- **Accuracy (Précision globale)** : Proportion des prédictions correctes parmi l'ensemble des données.
- **Precision (Précision par classe)** : Nombre de vrais positifs divisé par la somme des vrais positifs et des faux positifs. Indique la fiabilité des prédictions positives.
- **Recall (Rappel)** : Nombre de vrais positifs divisé par la somme des vrais positifs et des faux négatifs. Indique la capacité du modèle à détecter les échantillons positifs.

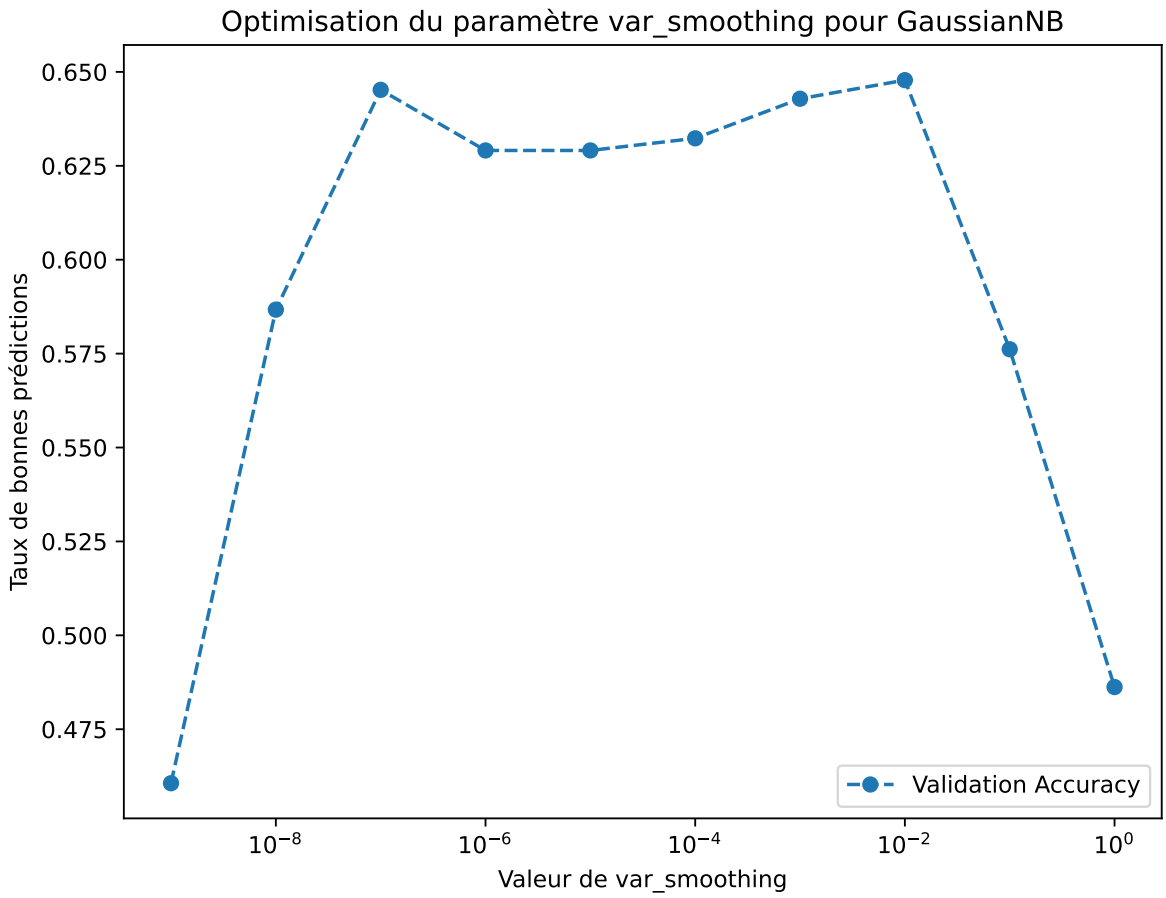
- **F1-score** : Moyenne harmonique entre précision et rappel, utile lorsque les classes sont déséquilibrées.

Exemple en Python

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split

# Chargement des ensembles de données déjà préparés
train_data = pd.read_csv('covertype_train.csv')
val_data = pd.read_csv('covertype_val.csv')
test_data = pd.read_csv('covertype_test.csv')

# Préparation des données
X_train = train_data.drop('Cover_Type', axis=1)
y_train = train_data['Cover_Type']

X_val = val_data.drop('Cover_Type', axis=1)
y_val = val_data['Cover_Type']

X_test = test_data.drop('Cover_Type', axis=1)
y_test = test_data['Cover_Type']

# Recherche du meilleur var_smoothing
var_smoothing_values = np.logspace(-9, 0, 10)
val_accuracies = []

for smoothing in var_smoothing_values:
    gnb = GaussianNB(var_smoothing=smoothing)
    gnb.fit(X_train, y_train)
    y_val_pred = gnb.predict(X_val)
    val_accuracies.append(accuracy_score(y_val, y_val_pred))

# Sélection du meilleur var_smoothing
best_smoothing = var_smoothing_values[val_accuracies.index(max(val_accuracies))]
print(f"Meilleur var_smoothing: {best_smoothing:.1e}")
```

```
# Affichage du graphique de performance
plt.figure(figsize=(8, 6))
plt.plot(var_smoothing_values, val_accuracies, marker='o', linestyle='dashed', label='Validat
plt.xscale('log')
plt.xlabel("Valeur de var_smoothing")
plt.ylabel("Taux de bonnes prédictions")
plt.title("Optimisation du paramètre var_smoothing pour GaussianNB")
plt.legend()
plt.show()

# Modèle final avec le meilleur hyperparamètre
gnb = GaussianNB(var_smoothing=best_smoothing)
gnb.fit(X_train, y_train)

# Évaluation sur l'ensemble de test
y_test_pred = gnb.predict(X_test)

# Affichage de la matrice de confusion avec annotations
conf_matrix = confusion_matrix(y_test, y_test_pred)
print("\nMatrice de confusion (les lignes représentent les vraies classes et les colonnes les
print(conf_matrix)

print("\nÉvaluation sur l'ensemble de test")
print(classification_report(y_test, y_test_pred))
```

```
Meilleur var_smoothing: 1.0e-02
```




```
Matrice de confusion (les lignes représentent les vraies classes et les colonnes les classes
[[1175  515    3    0    0    1   14]
 [ 512 1585  154    0    0    3    7]
 [   0   38  242    0    0    1    0]
 [   0    0   21    0    0    0    0]
 [   0   66    8    0    0    0    0]
 [   0   29  113    0    0    2    0]
 [ 150    0    0    0    0    0    9]]

Évaluation sur l'ensemble de test
              precision    recall  f1-score   support

           1       0.64      0.69      0.66      1708
           2       0.71      0.70      0.71      2261
           3       0.45      0.86      0.59       281
```

```
           4       0.00      0.00      0.00        21
           5       0.00      0.00      0.00        74
           6       0.29      0.01      0.03       144
           7       0.30      0.06      0.10       159

    accuracy                           0.65      4648
   macro avg       0.34      0.33      0.30      4648
weighted avg       0.63      0.65      0.63      4648


/home/ensai/.local/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1531: Unde
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/ensai/.local/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1531: Unde
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/home/ensai/.local/lib/python3.10/site-packages/sklearn/metrics/_classification.py:1531: Unde
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```