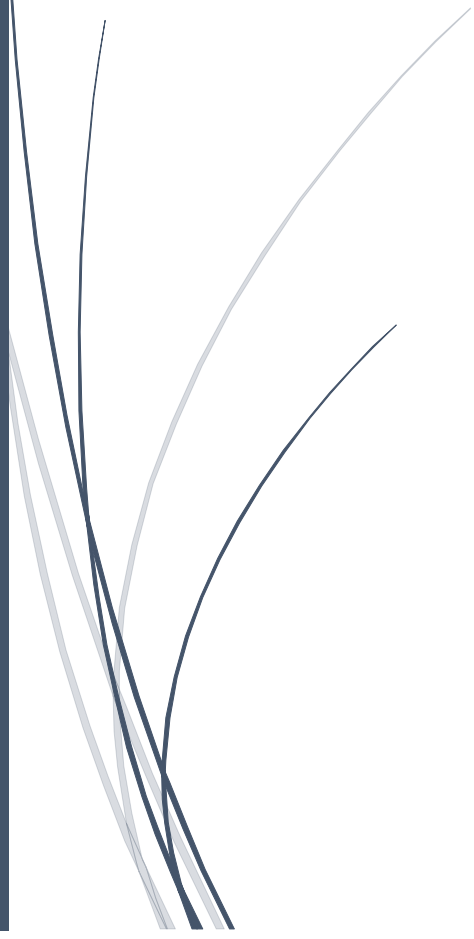
03/05/2023

# TP02 Naive Bayes

Machine Learning



Ramiqi Andi
MASTER IS

# Naive Bayes

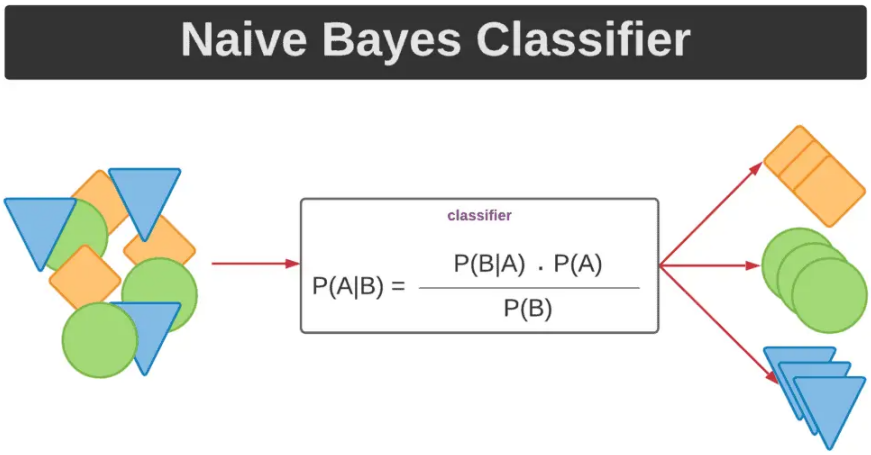
Naive Bayes utilise le théorème de Bayes et suppose que tous les prédicteurs sont indépendants. En d'autres termes, ce classificateur suppose que la présence d'une caractéristique particulière dans une classe n'affecte pas la présence d'une autre. Le terme “**naïve**” vient du fait qu’on suppose cette indépendance des variables.

**Avantages** :

- *Naive Bayes* est très rapide pour la classification : en effet les calculs de probabilités ne sont pas très coûteux.
- La classification est possible avec un petit ou grand jeu de données

**Inconvénients**

- L’algorithme Naive Bayes suppose l’indépendance des variables : C’est une hypothèse forte et qui est violée dans la majorité des cas réels




Le théorème de Bayes suppose le fait de transformer la probabilité à priori d’une classe y en probabilité à posteriori à l’aide des informations contenues dans l’observation X. Pour ce projet, la classe NaiveBayes() sera créée et permettra la gestion des différents types de données et des calculs de probabilités afin de classifier le jeu de test.

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y)P(y)}{P(\mathbf{x})}$$

$$= \frac{P(\mathbf{x}|y)P(y)}{\sum_{y'} P(\mathbf{x}|y')P(y')}$$

- P(y|X) est la probabilité postérieure qu'une instance appartienne à la classe Y, étant donné les caractéristiques observées X.
- P(X|y) est la vraisemblance(likelihood), il s'agit de la probabilité d'observer les caractéristiques X, étant donné la classe y.
- P(y) est la probabilité a priori de la classe y.
- P(X) est la probabilité des caractéristiques observées X.

# Code

Dans la classe NaiveBayes que nous mettons en place, la fonction « _calculate_posteriors » va calculer les probabilités postérieures pour chaque classe du vecteur de test après avoir mis en place l'entrainement avec la fonction « train » :

1. Pour les attributs continus, la fonction « _calculate_gaussian » est utilisée pour calculer la vraisemblance des données en supposant une distribution gaussienne. Le produit des probabilités conditionnelles est ensuite calculé pour chaque instance et chaque classe.

2. Pour les attributs catégoriels, la probabilité conditionnelle P(Xi|y) est extraite à partir des fréquences calculées lors de l'entraînement avec la méthode « _calculate_frequencies » appelée dans la méthode « train ». Le produit des probabilités conditionnelles est ensuite calculé pour chaque instance et chaque classe.

3. Les probabilités postérieures sont calculées en multipliant les vraisemblances (pour les attributs continus et catégoriels) par les probabilités a priori de chaque classe. Dans ce code, le terme P(X) est ignoré car il n'affecte pas la classification.

4. Finalement, la fonction « predict » utilise les probabilités postérieures calculées pour classifier chaque instance en choisissant la classe qui maximise la probabilité postérieure.

Afin de pouvoir expérimenter NaiveBayes, l'algorithme sera mis en place et utilisé pour quatre jeux de données différents :

1. Iris dataset
2. Breast cancer dataset
3. Handwritten digits dataset
4. Forest covertypes dataset

# Résultats :

| Dataset | Train Accuracy | Test Accuracy |
|---|---|---|
| Iris | 96.00% | 96.00% |
| Breast_cancer | 91.73% | 94.12% |
| Digits | 89.43% | 85.93% |
| Forest covertypes | 63.55% | 63.65% |

1. Le modèle Naïve Bayes fonctionne très bien sur le dataset Iris, avec une « accuracy » d'entraînement de 96% et une « accuracy » de test de 96%. Les performances sont assez élevées et il n'y a pas de surapprentissage ou de sous-apprentissage apparent. Cela est principalement dû à la simplicité du problème et à la séparabilité relativement bonne des classes.
2. Pour Breast_cancer, les performances sont aussi globalement bonnes et il ne semble pas y avoir de surapprentissage apparent.
3. Bien que les performances ne soient pas aussi élevées que pour les deux premiers datasets, le modèle semble tout de même bien fonctionner sur ce problème. La différence entre les « accuracy » d'entraînement et de test n'est pas trop importante, ce qui suggère que le modèle généralise relativement bien. Il serait tout à fait envisageable d'utiliser d'autres modèles afin d'améliorer les performances en utilisant par exemple des réseaux de neurones convolutionnels, qui sont plus adaptés aux problèmes de reconnaissance d'image.
4. Les performances du modèle pour « Forest covertypes » sont plus faibles avec une « accuracy » d'entraînement de 63,55% et une accuracy de test de 63,65%. Cela peut être principalement dû à la complexité du problème, au bruit dans les données, ou à un manque de séparabilité entre les classes. Une amélioration possible consisterait à explorer différentes techniques de prétraitement des données, de sélection des « features » ou de réduction de la dimensionnalité pour améliorer les performances du modèle.

# Decision surfaces

Comme nous pouvons le voir sur les trois surfaces de décisions différentes ci-dessous, celle-ci sont généralement linéaires en raison de l'hypothèse d'indépendance conditionnelle des caractéristiques. Ces surfaces permettent aussi d'observer les différentes séparation des classes et peuvent indiquer que certaines combinaisons d'attributs sont moins efficaces pour distinguer les classes. La troisième image présente néanmoins une distinction très nette des classes.

Ce genre de model permet ainsi de déterminer quelles caractéristiques sont les plus importantes pour la classification selon Naive Bayes. Malgré son hypothèse simplificatrice, Naive Bayes fonctionne ainsi plutôt bien dans la pratique, surtout pour un jeu de donnée limité tel que Iris.