Naive Bayes

Naive Bayes adalah salah satu algoritma pembelajaran mesin berbasis probabilistik yang sederhana namun sangat efektif, khususnya dalam klasifikasi teks, seperti analisis sentimen, deteksi spam, dan lain sebagainya. Algoritma ini didasarkan pada **Teorema Bayes**, yang merupakan dasar dari statistik Bayesian.

1. Dasar Teori

Teorema Bayes memberikan cara untuk menghitung probabilitas bersyarat dari suatu peristiwa berdasarkan informasi sebelumnya. Rumus Teorema Bayes adalah:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

- **P(A|B)**: Probabilitas hipotesis AAA terjadi, mengingat data BBB (posterior probability).
- **P(B|A)**: Probabilitas data BBB terjadi jika hipotesis AAA benar (likelihood).
- **P(A)**: Probabilitas awal hipotesis AAA (prior probability).
- **P(B)**: Probabilitas data BBB (evidence).

2. Asumsi Naive (Sederhana)

Naive Bayes disebut "naive" karena algoritma ini mengasumsikan bahwa semua fitur dalam data **saling independen** satu sama lain. Dengan kata lain, kontribusi setiap fitur terhadap hasil akhir dianggap tidak dipengaruhi oleh fitur lainnya. Walaupun asumsi ini jarang benar dalam kasus nyata, algoritma ini tetap bekerja dengan baik dalam banyak aplikasi.

3. Variasi Naive Bayes

Ada beberapa variasi Naive Bayes berdasarkan distribusi data yang diasumsikan:

- **Gaussian Naive Bayes**: Digunakan untuk data numerik yang diasumsikan mengikuti distribusi normal (Gaussian).
- **Multinomial Naive Bayes**: Cocok untuk data diskret, seperti frekuensi kata dalam teks.
- **Bernoulli Naive Bayes**: Digunakan untuk data biner, misalnya kehadiran atau ketidakhadiran fitur tertentu.

4. Cara Kerja Naive Bayes

Langkah-langkah utama dalam Naive Bayes:

1. **Menghitung Probabilitas Prior**:
    - Probabilitas awal untuk setiap kelas dalam dataset.

$$P(C) = \frac{\text{Jumlah contoh dalam kelas } C}{\text{Total jumlah contoh}}$$

2. **Menghitung Likelihood**:
    - Untuk setiap fitur, menghitung probabilitas kemunculannya dalam setiap kelas.
    - Formula:

$$P(F_i|C) = \frac{\text{Jumlah contoh dengan fitur } F_i \text{ dalam kelas } C}{\text{Jumlah total contoh dalam kelas } C}$$

3. **Menggabungkan Probabilitas**:
    - Dengan asumsi independensi, menghitung probabilitas gabungan dengan mengalikan setiap probabilitas fitur:

$$P(C|F_1, F_2, ..., F_n) \propto P(C) \cdot P(F_1|C) \cdot P(F_2|C) \cdot ... \cdot P(F_n|C)$$

4. **Prediksi**:
    - Memilih kelas dengan probabilitas tertinggi sebagai hasil prediksi.

5. Keunggulan Naive Bayes

- **Cepat dan Efisien**: Dapat menangani dataset besar karena perhitungan probabilitas relatif sederhana.
- **Efektif untuk Teks**: Sangat populer dalam klasifikasi teks, seperti deteksi spam email dan analisis sentimen.
- **Mudah Diimplementasikan**: Algoritma ini mudah dipahami dan diterapkan.

6. Kelemahan Naive Bayes

- **Asumsi Independensi**: Performanya bisa menurun jika fitur-fitur dalam data sangat bergantung satu sama lain.
- **Masalah Zero Frequency**: Jika suatu nilai fitur tidak muncul dalam data pelatihan, probabilitasnya akan menjadi nol. Ini dapat diatasi dengan teknik smoothing, seperti **Laplace Smoothing**:

$$P(F_i|C) = \frac{\text{Jumlah contoh dengan fitur } F_i + 1}{\text{Jumlah total contoh dalam kelas } C + \text{Jumlah total fitur}}$$

- **Tidak Cocok untuk Data Kontinu tanpa Distribusi Gaussian**: Jika data kontinu tidak mengikuti distribusi normal, algoritma ini bisa kurang efektif.

7. Contoh Aplikasi

1. **Deteksi Spam**: Mengklasifikasikan email sebagai spam atau bukan spam berdasarkan kata-kata yang muncul dalam email.
2. **Analisis Sentimen**: Menentukan apakah sebuah ulasan (review) bernada positif, negatif, atau netral.

3. **Prediksi Klasifikasi Penyakit**: Dalam bidang medis untuk memprediksi penyakit berdasarkan gejala pasien.
4. **Sistem Pencarian Informasi**: Membantu menyarankan artikel atau dokumen yang relevan.

8. Implementasi Naive Bayes dalam Python

Berikut contoh sederhana menggunakan scikit-learn:

```
import pandas as pd
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# 1. Membaca data dari file CSV
# Pastikan file CSV memiliki kolom 'text' untuk teks dan 'label'
untuk label
file_path = "data.csv"  # Ganti dengan nama file CSV Anda
data = pd.read_csv(file_path)

# 2. Melihat data untuk memastikan formatnya benar
print(data.head())

# 3. Memisahkan teks dan label
texts = data['text']  # Kolom yang berisi teks
labels = data['label']  # Kolom yang berisi label (misalnya, 0:
spam, 1: bukan spam)

# 4. Preprocessing teks menggunakan CountVectorizer
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(texts)  # Mengubah teks menjadi fitur
numerik
y = labels

# 5. Membagi data menjadi data latih dan data uji
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# 6. Melatih model Naive Bayes
model = MultinomialNB()
model.fit(X_train, y_train)

# 7. Melakukan prediksi pada data uji
y_pred = model.predict(X_test)
```

```
# 8. Evaluasi model
accuracy = accuracy_score(y_test, y_pred)
print("Akurasi model:", accuracy)

# 9. Contoh prediksi untuk teks baru
new_texts = ["penawaran menarik produk diskon besar", "cek email
penting ini"]
new_texts_vectorized = vectorizer.transform(new_texts)
predictions = model.predict(new_texts_vectorized)
print("Prediksi:", predictions)  # 0 untuk spam, 1 untuk bukan
spam
```

Penjelasan Langkah-Langkah

1. **Membaca Data**:
    - File .csv dibaca menggunakan pandas dengan pd.read_csv(). Pastikan file memiliki kolom untuk teks dan label.
2. **Preprocessing Teks**:
    - Menggunakan CountVectorizer untuk mengubah teks menjadi representasi numerik berbasis frekuensi kata.
3. **Pembagian Data**:
    - Membagi data menjadi data pelatihan (80%) dan data pengujian (20%) menggunakan train_test_split().
4. **Training Model**:
    - Menggunakan MultinomialNB untuk melatih model dengan data pelatihan.
5. **Prediksi**:
    - Model memprediksi kelas pada data pengujian dan teks baru.
6. **Evaluasi**:
    - Mengukur akurasi model menggunakan accuracy_score.

9. Kesimpulan

Naive Bayes adalah algoritma sederhana dan efisien yang cocok untuk berbagai masalah klasifikasi, terutama di bidang pengolahan teks. Meskipun asumsi independensinya sering kali tidak realistis, algoritma ini tetap menghasilkan performa yang baik dalam banyak aplikasi dunia nyata.