Nama : Akmal Zuhdy Prasetya
NIM : H071191035
Kelas : Machine Learning

Resume Pertemuan 5

A. Naive Bayesian

Algoritma Naive Bayes adalah sekumpulan algoritma pengklasifikasian dalam Supervised Machine Learning yang berdasarkan teorema probabilitas Bayes. Algoritma Naive Bayes mengasumsikan bahwa tidak ada korelasi antara fitur dalam dataset yang digunakan untuk melatih model. Pengklasifikasian Naive Bayes bekerja sangat baik dalam banyak masalah dunia nyata yang kompleks. Keuntungan besar dari pengklasifikasian Naive Bayes adalah ia hanya memerlukan sejumlah kecil sampel data pelatihan untuk melakukan klasifikasi secara efisien, dibandingkan dengan algoritme lain seperti Logistic Regression, Decision Trees, dan Support Vector Machines. Berikut beberapa istilah terkait dengan Naive Bayesian:

1. Teorema Bayes

   Teorema Bayes merupakan teorema yang menggambarkan probabilitas fitur, berdasarkan pengetahuan sebelumnya tentang situasi yang terkait dengan fitur itu. Misalnya, jika probabilitas seseorang menderita diabetes dikaitkan dengan usianya, maka dengan menggunakan teorema Bayes, usia dapat digunakan untuk memprediksi probabilitas diabetes secara lebih akurat.

2. Naive

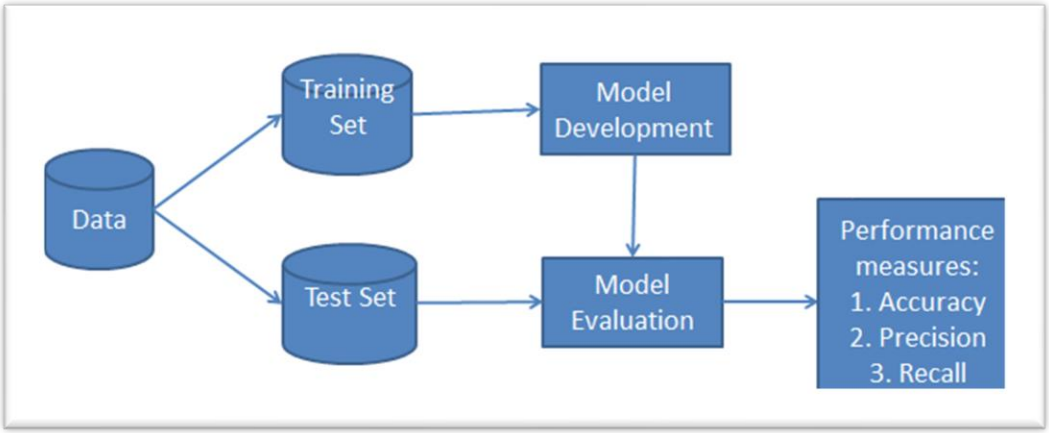
   Kata naif menyiratkan bahwa setiap pasangan fitur dalam kumpulan data independen satu sama lain. Semua pengklasifikasi Naive Bayes bekerja dengan asumsi bahwa nilai fitur tertentu tidak tergantung pada nilai fitur lain untuk kelas tertentu. Misalnya, buah dapat diklasifikasikan sebagai jeruk jika bulat, berdiameter sekitar 8 cm, dan berwarna oranye. Dengan pengklasifikasian Naive Bayes, masing-masing dari tiga fitur ini (bentuk, ukuran, dan warna) berkontribusi secara independen terhadap kemungkinan bahwa buah ini adalah jeruk. Selain itu, diasumsikan bahwa tidak ada kemungkinan korelasi antara atribut bentuk, ukuran, dan warna.

B. Naive Bayesian Workflow

Setiap kali kita melakukan klasifikasi, langkah pertama adalah memahami masalah dan mengidentifikasi fitur dan label. Fitur adalah karakteristik atau atribut yang mempengaruhi hasil label. Misalnya, dalam hal penyaluran pinjaman, manajer bank mengidentifikasi pekerjaan nasabah, pendapatan, usia, lokasi, riwayat pinjaman sebelumnya, riwayat transaksi, dan nilai kredit. Karakteristik ini dikenal sebagai fitur yang membantu model mengklasifikasikan pelanggan.

Klasifikasi ini memiliki dua fase, fase pembelajaran, dan fase evaluasi. Pada fase pembelajaran, classifier melatih modelnya pada dataset yang diberikan dan pada fase evaluasi, menguji kinerja classifier. Kinerja dievaluasi berdasarkan berbagai parameter seperti akurasi, kesalahan, presisi, dan ingatan.

Berikut adalah visualisasi dari workflow Naive Bayesian Classifier.



Naive Bayes adalah teknik klasifikasi statistik berdasarkan Teorema Bayes. Ini adalah salah satu algoritma pembelajaran terawasi yang paling sederhana. Pengklasifikasi Naive Bayes adalah algoritma yang cepat, akurat dan dapat diandalkan. Pengklasifikasi Naive Bayes memiliki akurasi dan kecepatan tinggi pada kumpulan data yang besar.

Pengklasifikasi Naive Bayes mengasumsikan bahwa efek dari fitur tertentu dalam suatu kelas tidak tergantung pada fitur lainnya. Misalnya, pemohon pinjaman diinginkan atau tidak tergantung pada pendapatannya, pinjaman sebelumnya dan riwayat transaksi, usia, dan lokasi. Sekalipun fitur-fitur ini saling bergantung, fitur-fitur ini masih dianggap independen. Asumsi ini menyederhanakan perhitungan, dan karena itu dianggap naif. Asumsi ini disebut independensi bersyarat kelas. Berikut adalah bentuk matematis dari peritungan Naive Bayesian meggunakan Teorema Bayes:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

Dengan keterangan:

- $P(h)$: probabilitas hipotesis h bernilai benar (terlepas dari datanya). Ini dikenal sebagai probabilitas prior dari h.
- $P(D)$: probabilitas data (terlepas dari hipotesis). Ini dikenal sebagai probabilitas prior.
- $P(h|D)$: probabilitas hipotesis h berdasarkan data D. Ini dikenal sebagai probabilitas posterior.
- $P(D|h)$: probabilitas data D jika hipotesis h benar. Ini dikenal sebagai probabilitas posterior.

Classifier Naive Bayes menghitung probabilitas suatu peristiwa dalam langkah-langkah berikut:

1. Hitung probabilitas sebelumnya untuk label kelas yang diberikan.
2. Temukan probabilitas Kemungkinan dengan setiap atribut untuk setiap kelas.
3. Masukkan nilai ini ke dalam Rumus Bayes dan hitung probabilitas posterior.
4. Lihat kelas mana yang memiliki probabilitas lebih tinggi, mengingat input milik kelas probabilitas yang lebih tinggi.

C. Implementasi Naive Bayesian

Berikut merupakan contoh implementasi dari Classifier Naive Bayesian dengan dataset yang memiliki detail sebagai berikut:

- Pregnancies: Jumlah berapa kali hamil
- Glucose: Konsentrasi glukosa plasma 2 jam dalam tes toleransi glukosa oral
- BloodPressure: Tekanan darah diastolik (mm Hg)
- SkinThickness: Ketebalan lipatan kulit trisep (mm)
- Insulin: insulin serum 2 Jam (mu U/ml)
- BMI: Indeks massa tubuh (berat dalam kg/(tinggi dalam m)^2)
- DiabetesPedigreeFunction: Fungsi silsilah diabetes
- Age: Usia (tahun)
- Outcome: Variabel kelas (0 atau 1), menentukan apakah menderita diabetes atau tidak

Sumber dataset: https://www.kaggle.com/mathchi/diabetes-data-set

1. Import Library

```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

2. Import Dataset

```
os.environ['KAGGLE_USERNAME'] = "akmalzuhdyprasetya"
os.environ['KAGGLE_KEY'] = "3f1fd1ffa5d36294293dfb098b3b7392"

!kaggle datasets download -d mathchi/diabetes-data-set

!unzip -q /content/diabetes-data-set.zip

data = pd.read_csv("/content/diabetes.csv")
data.head(5)
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

3. Data Pre-Processing

Sebelum memasukkan data ke model classifier Naive Bayes, kita perlu melakukan beberapa pra-pemrosesan. Di sini, kita akan membuat variabel `x` dan `y` dengan mengambilnya dari dataset dan menggunakan method `.train_test_split()` dari `scikit-learn` untuk membagi data menjadi dataset untuk pelatihan dan pengujian. Perlu diperhatikan bahwa ukuran pengujian 0.25 menunjukkan bahwa kita telah menggunakan 25% data untuk pengujian, parameter `random_state` memastikan tingkat reproduktifitas. Untuk output `.train_test_split()`, kita mendapatkan nilai `x_train`, `x_test`, `y_train`, dan `y_test`.

```
x = data.drop('Outcome', axis=1)
y = data['Outcome']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.50,
                                                    random_state=42)
```

a. `x_train`

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
727	0	141	84	26	0	32.4	0.433	22
423	2	115	64	22	0	30.8	0.421	21
179	5	130	82	0	0	39.1	0.956	37
304	3	150	76	0	0	21.0	0.207	37
398	3	82	70	0	0	21.1	0.389	25
...	...	...	...	...	...	...	...	...
71	5	139	64	35	140	28.6	0.411	26
106	1	96	122	0	0	22.4	0.207	27
270	10	101	86	37	0	45.6	1.136	38
435	0	141	0	0	0	42.4	0.205	29
102	0	125	96	0	0	22.5	0.262	21

384 rows × 8 columns

b. `x_test`

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
668	6	98	58	33	190	34.0	0.430	43
324	2	112	75	32	0	35.7	0.148	21
624	2	108	64	0	0	30.8	0.158	21
690	8	107	80	0	0	24.6	0.856	34
473	7	136	90	0	0	29.9	0.210	50
...	...	...	...	...	...	...	...	...
578	10	133	68	0	0	27.0	0.245	36
664	6	115	60	39	0	33.7	0.245	40
100	1	163	72	0	0	39.0	1.222	33
445	0	180	78	63	14	59.4	2.420	25
689	1	144	82	46	180	46.1	0.335	46

384 rows × 8 columns

c. `y_train`

```
727    0
423    0
179    1
304    0
398    0
      ..
71     0
106    0
270    1
435    1
102    0
Name: Outcome, Length: 384, dtype: int64
```

d. `y_test`

```
668    0
324    0
624    0
690    0
473    0
      ..
578    0
664    1
100    1
445    1
689    1
Name: Outcome, Length: 384, dtype: int64
```

4. Model Training

Kita akan menggunakan `x_train` dan `y_train` yang telah diperoleh, untuk melatih model classifier Naive Bayes. Kita akan menggunakan method `.fit()` dan memberikan parameter seperti yang ditunjukkan di bawah ini. Perhatikan bahwa output dari sel ini menjelaskan beberapa parameter seperti `prior` dan `var_smoothing` untuk model. Semua parameter ini dapat dikonfigurasi, dan kita bebas menyetelnya agar sesuai dengan kebutuhan kita.

```
model = GaussianNB()
model.fit(x_train, y_train)
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

5. Prediction

Setelah model dilatih, model siap untuk membuat atau melakukan prediksi. Kita dapat menggunakan method `.predict()` pada model dan memberikan `x_test` sebagai parameter untuk mendapatkan output yaitu `y_pred`.

```
y_pred = model.predict(x_test)
```

a. `y_pred`

```
array([0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 0, 0, 0, 1, 1, 1])
```

Contoh hasil prediksi:

```
x_test.iloc[10]

Pregnancies                  10.000
Glucose                     111.000
BloodPressure                70.000
SkinThickness                27.000
Insulin                       0.000
BMI                          27.500
DiabetesPedigreeFunction      0.141
Age                          40.000
Name: 667, dtype: float64

y_pred[10]

0
```

6. Model Evaluation

Terakhir, kita perlu memeriksa untuk melihat seberapa baik kinerja model kita pada data uji. Untuk ini, kita mengevaluasi model kami dengan menemukan skor akurasi yang dihasilkan oleh model.

```
acc = accuracy_score(y_test, y_pred)*100

print('Model Accuracy', acc, sep=': ')

Model Accuracy: 76.30208333333334
```

Dapat dilihat bahwa model yang telah kita latih memiliki tingkat keakuratan sebesar 76%.

Source code: https://github.com/trueazp/Machine-Learning/blob/main/assignments/assignment04/Tugas04_ML_H071191035.ipynb