Laporan Tugas Pemrograman 3
Learning dengan Algoritma Naive Bayes

Dosen mata kuliah : Siti Sa'adah, ST., M.T.

Disusun oleh :
Famardi Putra Muhammad Raffly - 1301204391
Muhammad Daffa' Ibrahim - 1301204051

Fakultas Informatika
Telkom University
Bandung
2022

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Naïve Bayes Classifier merupakan sebuah metode klasifikasi yang berakar pada teorema Bayes . Metode pengklasifikasian dengan menggunakan metode probabilitas dan statistik yg dikemukakan oleh ilmuwan Inggris Thomas Bayes , yaitu memprediksi peluang di masa depan berdasarkan pengalaman di masa sebelumnya sehingga dikenal sebagai Teorema Bayes . Ciri utama dr Naïve Bayes Classifier ini adalah asumsi yg sangat kuat (naïf) akan independensi dari masing-masing kondisi / kejadian. Teorema Naive Bayes dirumuskan sebagai berikut.

$$P(A|B) = P(B|A)\ P(A)\ P(B)$$

P(A) : Peluang terjadinya A

P(B) : Peluang terjadinya B

P(A | B) : Probabilitas A terjadi dengan bukti B telah terjadi

P(B | A) : Probabilitas B terjadi dengan bukti bahwa A telah terjadi

Naive bayes adalah algoritma yang banyak digunakan untuk membuat model machine learning dan memiliki keunikannya sendiri. Algoritma ini memiliki beberapa tipe, berikut beberapa tipenya:

1. ***Bernoulli Naive Bayes***

   Pada algoritma ini, prediktor adalah variabel boolean. Dikarenakan prediktornya variabel boolean, maka satu-satunya nilai yang ada adalah benar atau salah. Algoritma ini digunakan ketika data sesuai dengan distribusi *bernoulli multivariat*.

2. ***Naive Bayes Multinomial***

   Algoritma ini digunakan untuk memecahkan masalah klasifikasi dokumen. Contohnya, apabila ingin menentukan apakah suatu dokumen termasuk dalam suatu kategori, algoritma ini dapat digunakan untuk memilahnya. *Naive bayes* menggunakan frekuensi kata-kata sekarang sebagai fitur.

3. ***Gaussian Naive Bayes***

   Algoritma ini digunakan apabila prediktor tidak diskrit namun memiliki nilai kontinu. Prediktor tersebut diasumsikan sebagai sampel dari distribusi gaussian.

1.2 Spesifikasi Tugas

Diberikan file traintest.xlsx yang terdiri dari dua sheet: train dan test, yang berisi dataset untuk problem klasifikasi biner (binary classification). Setiap record atau baris data dalam dataset tersebut secara umum terdiri dari nomor baris data (*id*), fitur input ($x1$ sampai $x3$), dan output kelas (y). Fitur input terdiri dari nilai-nilai integer dalam range tertentu untuk setiap fitur. Sedangkan output kelas bernilai biner (0 atau 1).

id	x1	x2	x3	y
1	60	64	0	1
2	54	60	11	0
3	65	62	22	0
4	34	60	0	1
5	38	69	21	0

Sheet train berisi 296 baris data, lengkap dengan target output kelas (y). Gunakan sheet ini untuk tahap pemodelan atau pelatihan (training) model sesuai metode yang Anda gunakan. Adapun sheet test berisi 10 baris data, dengan output kelas (y) yang disembunyikan. Gunakan sheet ini untuk tahap pengujian (testing) model yang sudah dilatih. Nantinya output program Anda untuk data uji ini akan dicocokkan dengan target atau kelas sesungguhnya.

id	x1	x2	x3	y
297	43	59	2	?
298	67	66	0	?
299	58	60	3	?
300	49	63	3	?
301	45	60	0	?
302	54	58	1	?
303	56	66	3	?
304	42	69	1	?
305	50	59	2	?
306	59	60	0	?

BAB 2
PEMBAHASAN

2.1 Membaca/Mengimport Data

```
IMPORT FILE

[ ]  #Mengimport data dari file bengkel.xlsx
     from google.colab import files
     TheFile = files.upload()

     Choose Files  No file chosen        Upload widget is only available when the
     Saving traintest.xlsx to traintest.xlsx

[ ]  data = pd.read_excel('traintest.xlsx' ,sheet_name = "train")
     datatest = pd.read_excel('traintest.xlsx' ,sheet_name = "test")
```

Berikut merupakan proses membaca atau mengimpor data traintest.xlsx yang berupa file excel menjadi sebuah array pada python. array “data” merupakan data train yang akan menjadi bahan dasar untuk mencari nilai prediksi dan array “datatest” yang merupakan data test yang akan dicari nilai prediksinya.

2.2 Membangun Model

```
dataTrain = np.array(data)

latih = int((0.8 * len(dataTrain)))
dataUji =[]
dataLatih = dataTrain[:int(latih)]
dataUji = dataTrain[int(latih): len(dataTrain)]

tabelData = pd.DataFrame(dataTrain, columns =['id', 'x1', 'x2', 'x3', 'y'])
print(tabelData)
```

Dalam proses membangun model, metode validasi yang kami gunakan adalah metode Holdout Validation dengan rasio perbandingan data latih dan data uji adalah 80% banding 20%. Kami membagi 80% data awal pada data train menjadi data latih dan 20% data terakhir pada data train menjadi data uji.

```
idYes = []
x1Yes = []
x2Yes = []
x3Yes = []
yYes = []
idNo = []
x1No = []
x2No = []
x3No = []
yNo = []

for i in range(len(dataLatih)):
  if dataLatih[i][4] == 1:
    idYes.append(dataLatih[i][0])
    x1Yes.append(dataLatih[i][1])
    x2Yes.append(dataLatih[i][2])
    x3Yes.append(dataLatih[i][3])
    yYes.append(dataLatih[i][4])
  else:
    idNo.append(dataLatih[i][0])
    x1No.append(dataLatih[i][1])
    x2No.append(dataLatih[i][2])
    x3No.append(dataLatih[i][3])
    yNo.append(dataLatih[i][4])
```

```
mean_X1Yes = mean(x1Yes)
mean_X2Yes = mean(x2Yes)
mean_X3Yes = mean(x3Yes)

mean_X1No = mean(x1No)
mean_X2No = mean(x2No)
mean_X3No = mean(x3No)

Var_X1Yes = variansi(x1Yes)
Var_X2Yes = variansi(x2Yes)
Var_X3Yes = variansi(x3Yes)

Var_X1No = variansi(x1No)
Var_X2No = variansi(x2No)
Var_X3No = variansi(x3No)
```

Proses selanjutnya adalah memecah data berdasarkan nilai y nya, seluruh kesatuan data akan di assign menjadi array baru sesuai dengan nilai y nya. Data x1 yang memiliki nilai y = 1 akan di assign pada array x1Yes, sedangkan data x1 yang memiliki nilai y = 0 akan di assign pada array x1No, data x2 yang memiliki nilai y = 1 akan di assign pada array x2Yes, sedangkan data x2 yang memiliki nilai y = 0 akan di assign pada array x2No, dan data x3 yang memiliki nilai y = 1 akan di assign pada array x3Yes, sedangkan data x3 yang memiliki nilai y = 0 akan di assign pada array x3No, begitupun dengan data id dan data y. Kemudian mencari nilai rata-rata dan nilai variansi pada array x1Yes, x2Yes, x3Yes, x1No, x2No, dan x3 No. Berikut merupakan proses mencari nilai rata-rata dan nilai variansi:

HITUNG MEAN

```
def mean(nilai):
  jumlah = 0
  for i in range(len(nilai)):
    jumlah = jumlah + nilai[i]

  rata2 = jumlah / len(nilai)

  return rata2
```

HITUNG VARIANSI

```
[6] def variansi(nilai):
      rata2 = mean(nilai)
      jumlah = 0
      for i in range(len(nilai)):
        jumlah = (jumlah + ((nilai[i] - rata2)**2))

      hasil = (jumlah / (len(nilai)-1))

      return math.sqrt(hasil)
```

2.3 Gaussian Naive Bayes

Gaussian Naive Bayes adalah sebuah varian dari metode naive bayes yang mengikuti distribusi normal gaussian dan bisa digunakan pada data bertipe kontinu atau numerik. Data kontinu dapat berupa data retensi mahasiswa yang memiliki variabel nilai IPK atau jumlah mata kuliah yang sedang diambil. Gaussian naive bayes ini juga digunakan untuk menghitung nilai prediksi suatu data.

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

(X_i | y) = nilai (X_i | y)
σ = Standar Deviasi dari atribut (X_i|y)
μ = mean atau rata-rata dari (X_i|y)

NAIVE BAYES

```
def bayes(x, mean,var):
  hasil = 1 / (var * math.sqrt(2 * 3.14)) * math.exp(-((x - mean)**2 / (2 * var**2)))

  return hasil
```

2.4 Memprediksi Data Uji

```
[ ] TheprediksiUji = []

    for i in range(len(dataUji)):
      yes = len(idYes)/len(dataLatih) * (bayes(dataUji[i][1], mean_X1Yes, Var_X1Yes)) * (bayes(dataUji[i][2], mean_X2Yes, Var_X2Yes)) * (bayes(dataUji[i][3], mean_X3Yes, Var_X3Yes))
      no = len(idNo)/len(dataLatih) * (bayes(dataUji[i][1], mean_X1No, Var_X1No)) * (bayes(dataUji[i][2], mean_X2No, Var_X2No)) * (bayes(dataUji[i][3], mean_X3No, Var_X3No))

      if yes > no:
        TheprediksiUji.append(1)
      else:
        TheprediksiUji.append(0)
```

Menghitung nilai prediksi dari data uji dan menampung hasil prediksi tersebut pada array TheprediksiUji. Akan dihitung peluang nilai y = 1 dan nilai peluang nilai y = 0 dengan menggunakan teorema naive, kemudian jika peluang nilai y = 1 lebih besar, maka akan menambah nilai 1 pada array TheprediksiUji, sebaliknya jika peluang nilai y = 0 lebih besar, maka akan menambah nilai 0 pada array TheprediksiUji.

2.5 Menghitung Akurasi Data Uji

MENGHITUNG AKURASI PADA DATA UJI

```
tpuji = 0
fnuji = 0
fpuji = 0
tnuji = 0

for i in range (len(dataUji)):
  if dataUji[i][4] == 1 and TheprediksiUji[i] == 1:
    tpuji = tpuji + 1
  elif dataUji[i][4] == 1 and TheprediksiUji[i] == 0:
    fnuji = fnuji + 1
  elif dataUji[i][4] == 0 and TheprediksiUji[i] == 1:
    fpuji = fpuji + 1
  elif dataUji[i][4] == 0 and TheprediksiUji[i] == 0:
    tnuji = tnuji + 1

akurasiUji = (tpuji + tnuji) / (tpuji + tnuji + fnuji + fpuji)
```

```
print("AKURASI DATA UJI : ",akurasiUji * 100, "%")
```

```
AKURASI DATA UJI :  75.0 %
```

Menghitung nilai akurasi pada data uji menggunakan confusion matrix dengan mencari terlebih dahulu nilai true positif, true negatif, false positif, dan false negatif dari hasil perbandingan nilai y pada array data uji dengan nilai pada array TheprediksiUji. Didapatkan hasil akurasi data uji sebesar 75%.

2.6 Memprediksi Data Keseluruhan

```
TheprediksiSeluruh = []

for i in range(len(dataTrain)):
  yes = len(idYes)/len(dataLatih) * (bayes(dataTrain[i][1], mean_X1Yes, Var_X1Yes)) * (bayes(dataTrain[i][2], mean_X2Yes, Var_X2Yes)) * (bayes(dataTrain[i][3], mean_X3Yes, Var_X3Yes))
  no = len(idNo)/len(dataLatih) * (bayes(dataTrain[i][1], mean_X1No, Var_X1No)) * (bayes(dataTrain[i][2], mean_X2No, Var_X2No)) * (bayes(dataTrain[i][3], mean_X3No, Var_X3No))

  if yes > no:
    TheprediksiSeluruh.append(1)
  else:
    TheprediksiSeluruh.append(0)
```

Menghitung nilai prediksi dari data keseluruhan dan menampung hasil prediksi tersebut pada array TheprediksiSeluruh. Akan dihitung peluang nilai y = 1 dan nilai peluang nilai y = 0 dengan menggunakan teorema naive, kemudian jika peluang nilai y = 1 lebih besar, maka akan menambah nilai 1 pada array TheprediksiSeluruh, sebaliknya jika peluang nilai y = 0 lebih besar, maka akan menambah nilai 0 pada array TheprediksiSeluruh.

2.7 Menghitung Akurasi Data Keseluruhan

MENGHITUNG AKURASI PADA DATA KESELURUHAN

```
tp = 0
fn = 0
fp = 0
tn = 0

for i in range (len(dataUji)):
  if dataTrain[i][4] == 1 and TheprediksiSeluruh[i] == 1:
    tp = tp + 1
  elif dataTrain[i][4] == 1 and TheprediksiSeluruh[i] == 0:
    fn = fn + 1
  elif dataTrain[i][4] == 0 and TheprediksiSeluruh[i] == 1:
    fp = fp + 1
  elif dataTrain[i][4] == 0 and TheprediksiSeluruh[i] == 0:
    tn = tn + 1

akurasiKeseluruhan = (tp + tn) / (tp + tn + fn + fp)
```

```
[ ] print("AKURASI DATA KESELURUHAN : ",akurasiKeseluruhan * 100, "%")
```

```
AKURASI DATA KESELURUHAN :  80.0 %
```

Menghitung nilai akurasi pada data keseluruhan menggunakan confusion matrix dengan mencari terlebih dahulu nilai true positif, true negatif, false positif, dan false negatif dari hasil perbandingan nilai y pada array data uji dengan nilai pada array TheprediksiSeluruh. Didapatkan hasil akurasi data uji sebesar 80%.

2.8 Hasil Data Testing

```
[17] dataTest = np.array(datatest)

finalprediksi = []
for i in range(len(dataTest)):
  yes = len(idYes)/len(dataLatih) * (bayes(dataTest[i][1], mean_X1Yes, Var_X1Yes)) * (bayes(dataTest[i][2], mean_X2Yes, Var_X2Yes)) * (bayes(dataTest[i][3], mean_X3Yes, Var_X3Yes))
  no = len(idNo)/len(dataLatih) * (bayes(dataTest[i][1], mean_X1No, Var_X1No)) * (bayes(dataTest[i][2], mean_X2No, Var_X2No)) * (bayes(dataTest[i][3], mean_X3No, Var_X3No))

  if yes > no:
    dataTest[i][4] = 1
  else:
    dataTest[i][4] = 0
```

Menghitung nilai prediksi dari data testing, akan dihitung peluang nilai y = 1 dan nilai peluang nilai y = 0 dengan menggunakan teorema naive, kemudian jika peluang nilai y = 1 lebih besar, maka nilai y pada array data testing akan di assign = 1, sebaliknya jika peluang nilai y = 0 lebih besar, maka nilai y pada array data testing akan di assign = 0. Berikut merupakan hasil prediksi dari data testing:

```
[ ] result = pd.DataFrame(dataTest, columns =['id', 'x1', 'x2', 'x3', 'y'])
    print(result)

    #Mengubah menjadi file excel
    result.to_excel('TugasPemrograman3.xlsx')
```

```
    id  x1  x2 x3  y
0  297  43  59  2  1
1  298  67  66  0  1
2  299  58  60  3  1
3  300  49  63  3  1
4  301  45  60  0  1
5  302  54  58  1  1
6  303  56  66  3  1
7  304  42  69  1  1
8  305  50  59  2  1
9  306  59  60  0  1
```

BAB 3
KESIMPULAN

Berdasarkan pengujian yang kami lakukan di atas, dimana kami menggunakan algoritma Naïve Bayes dan metode Holdout validation untuk membangun model dengan perbandingan pembagian data sebesar 80% untuk data training dan 20% untuk data uji, dapat disimpulkan bahwa perhitungan akurasi yang didapat untuk data uji cukup tinggi, yaitu sebesar 75%. Kemudian, kami menghitung juga akurasi untuk data keseluruhan, dimana akurasi yang didapat dari hasil perhitungan data keseluruhan lebih besar daripada akurasi data uji, yaitu sebesar 80%. Hasil akurasi yang didapat akan berbeda-beda tergantung pada penggunaan algoritma, metode validation yang digunakan dan berhubung kami menggunakan metode Holdout Validation, maka rasio perbandingan data latih dan data uji tentu akan mempengaruhi hasil akurasi. Model yang telah dibangun dari hasil pembelajaran pada data training dapat digunakan untuk memprediksi data lain apabila akurasi yang didapat cukup tinggi. Dapat disimpulkan, model yang kami buat di atas dapat digunakan untuk memprediksi data lain karena memiliki akurasi yang cukup tinggi. Dengan menggunakan algoritma dan metode yang tepat, akan didapat hasil prediksi dan akurasi yang maksimal.

LAMPIRAN

Link video :
https://drive.google.com/file/d/15A2chnti9GFGqn4J_ZdHT3qDp5J9n-1I/view?usp=sharing
Link Collab :
https://colab.research.google.com/drive/17v6CwErCyAXRsjyfrcPtBNmTS921FEsy?usp=sharing

PEMBAGIAN JOBDESK

Nama Mahasiswa	Job Desk	Kontribusi
Famardi Putra Muhammad Raffly	Membuat program dan laporan	50%
Muhammad Daffa' Ibrahim	Membuat program dan laporan	50%