Laporan Tugas Pemrograman 3 Learning dengan Algoritma Naive Bayes

Dibuat untuk memenuhi tugas pemrograman mata kuliah pengantar kecerdasan buatan

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

Disusun oleh :
Aisyah Dliya Ramadhanti (130120)
Fijar Yasmina Pritama (1301200215)

Fakultas Informatika
Universitas Telkom
Bandung
2022

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Naive bayes adalah suatu metode yang cocok digunakan untuk klasifikasi biner dan multiclass. Metode ini menerapkan teknik supervised klasifikasi objek dengan menetapkan label kelas menggunakan probabilitas bersyarat. Probabilitas bersyarat adalah nilai peluang suatu peristiwa yang terjadi berdasarkan peristiwa yang telah terjadi.

Supervised learning menunjukkan klasifikasi training data yang sudah diberi label dengan kelas. Salah satu contohnya yaitu, jika seorang melakukan transaksi penipuan di m-banking, maka transaksi itu sudah ditandai sebagai data transaksi. Lalu, apabila ingin mengklasifikasikan data transaksi menjadi transaksi sah atau transaksi tidak sah, maka jenis klasifikasi tersebut akan dinamakan supervised. Teorema Naive Bayes dirumuskan sebagai berikut.

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

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

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

P(A) : Peluang terjadinya A

P(B) : Peluang terjadinya B

Tujuan metode naive bayes adalah mengklasifikasikan probabilitas berdasarkan pembelajaran mesin atas probabilitas lain. Metode ini sudah banyak diterapkan di kehidupan sehari -hari seperti prakiraan cuaca. Program prakiraan cuaca dapat memprediksi cuaca hari besoknya seperti cerah, hujan, atau berangin berdasarkan suhu, kelembaban, tekanan udara, dan lain - lain. Selain itu, naive bayes juga memiliki beberapa classifier sebagai berikut :

1. Multinomial Naive Bayes

   Metode ini digunakan untuk mengklasifikasi data berbentuk dokumen. Dokumen - dokumen dapat dikategorikan berdasarkan temanya seperti olahraga, hiburan, ekonomi, atau berdasarkan kata yang sering muncul dalam dokumen.
2. Bernoulli Naive Bayes

   Metode Bernoulli bekerja seperti metode multinomial, tetapi klasifikasinya lebih berfokus pada hasil ya atau tidak. Inputan berupa variabel boolean. Contoh penerapannya yaitu prediksi kemunculan kata pada suatu teks.
3. Gaussian Naive Bayes

   Metode ini menggunakan distribusi Gaussian yang mendistribusi nilai

kontinu yang terkait dengan fitur berisi nilai numerik. Metode Gaussian akan mengeluarkan plot yang berbentuk lonceng simetris sesuai rata - rata nilai fitur.

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.

BAB 2
PEMBAHASAN

2.1 Membaca data latih/uji

MEMBACA DATA

```
[8]  import pandas as pd
     import matplotlib.pyplot as plt
     import numpy as np
     import math
     from google.colab import files
     up = files.upload()
     df_train = pd.read_excel("traintest (1).xlsx", sheet_name = "train")
     df_test = pd.read_excel("traintest (1).xlsx", sheet_name = "test")
```

Langkah awal pada tugas pemrograman 3 tentang learning ini adalah mengunggah dan membaca file excel traintest.xlsx. File ini berisi data latih dan data uji, yang akan menjadi bahan dasar dalam proses learning ini.

2.2 Klasifikasi Data Train

Data latih dan data uji yang digunakan untuk klasifikasi berjumlah 259 data latih`dan 37 data uji, kemudian data latih akam proses menggunakan metode naive bayes. Berikut merupakan data latih yang digunakan.

```
----DATA LATIH----
      ID  X1  X2  X3  y
0      1  60  64   0  1
1      2  54  60  11  0
2      3  65  62  22  0
3      4  34  60   0  1
4      5  38  69  21  0
..   ...  ..  ..  .. ..
254  255  65  58   0  1
255  256  83  58   2  0
256  257  34  61  10  1
257  258  51  66   1  1
258  259  61  62   5  0

[259 rows x 5 columns]
```

2.3 Membangun Model

```
dataTrain = np.array(df_train)

#MENDEKLARASIKAN ARRAY DATA LATIH DAN DATA UJI
dataLatih = []
dataUji = []

#DATA LATIH DAN DATA UJI
nAwal = 0.875 * len(dataTrain)

dataLatih = dataTrain[:int(nAwal)]
dataUji = dataTrain[int(nAwal): len(dataTrain)]

#MODEL DATA LATIH y = 1
modelYes = [[0,0], [0,0], [0,0]]
#MODEL DATA LATIH y = 0
modelNo =[[0,0], [0,0], [0,0]]

#MENDEKLARASI DATA LATIH y = 1
TrainYes = []
#MENDEKLARASI DATA LATIH y = 0
TrainNo = []
```

```
#DATA LATIH
for i in range(len(dataLatih)):
  if (dataLatih[i][4] == 1) :
    TrainYes.append(dataLatih[i])
  else:
    TrainNo.append(dataLatih[i])

#MEMBANGUN MODEL
modelYes[0][0] = mean_x1(TrainYes)
modelYes[0][1] = variansi_x1(TrainYes)
modelYes[1][0] = mean_x2(TrainYes)
modelYes[1][1] = variansi_x2(TrainYes)
modelYes[2][0] = mean_x3(TrainYes)
modelYes[2][1] = variansi_x3(TrainYes)

modelNo[0][0] = mean_x1(TrainNo)
modelNo[0][1] = variansi_x1(TrainNo)
modelNo[1][0] = mean_x2(TrainNo)
modelNo[1][1] = variansi_x2(TrainNo)
modelNo[2][0] = mean_x3(TrainNo)
modelNo[2][1] = variansi_x3(TrainNo)

print(modelYes)
print(modelNo)
```

Pada proses pemilihan model data, kami menggunakan holdout validation dengan perbandingan 87,5% : 12,5%. Setelah itu, kita akan membagi data latih

menjadi 2 bagian. Data latih dengan y = 1 atau y = 0. Kedua data yang sudah terbagi berdasarkan nilai y akan dimasukkan ke dua array yang berbeda, array data yes (TrainYes) dan array data latih no (TrainNo). Selanjutnya, akan dihitung mean dan variansi dari masing - masing data menggunakan fungsi di bawah ini.

```
#RATA-RATA DARI X1
def mean_x1(listTrain):
  sumx1 = 0
  for i in range(len(listTrain)):
    sumx1 = sumx1 + listTrain[i][1]
  return sumx1 / (len(listTrain))

#RATA-RATA DARI X2
def mean_x2(listTrain):
  sumx2 = 0
  for i in range(len(listTrain)):
    sumx2 = sumx2 + listTrain[i][2]
  return sumx2 / (len(listTrain))

#RATA-RATA DARI X3
def mean_x3(listTrain):
  sumx3 = 0
  for i in range(len(listTrain)):
    sumx3 = sumx3 + listTrain[i][3]
  return sumx3 / (len(listTrain))
```

```
def variansi_x1(listTrain):
  avg = mean_x1(listTrain)
  sumX1 = 0
  for i in range(len(listTrain)):
    sumX1 = (sumX1 +((listTrain[i][1] - avg)**2))

  resultX1 = (sumX1 / (len(listTrain)-1))
  return math.sqrt(resultX1)

#VARIANSI DARI X2
def variansi_x2(listTrain):
  avg = mean_x2(listTrain)
  sumX2 = 0
  for i in range(len(listTrain)):
    sumX2 = (sumX2 +((listTrain[i][2] - avg)**2))

  resultX2 = (sumX2 / (len(listTrain)-1))
  return math.sqrt(resultX2)

#VARIANSI DARI X3
def variansi_x3(listTrain):
  avg = mean_x3(listTrain)
  sumX3 = 0
  for i in range(len(listTrain)):
    sumX3 = (sumX3 +((listTrain[i][3] - avg)**2))

  resultX3 = (sumX3 / (len(listTrain)-1))
  return math.sqrt(resultX3)
```

Setelah menemukan nilai mean dan variansi, nilai tersebut akan dimasukkan ke array yang berisi nilai mean dan variansi dari model data y=1 (modelYes) dan model data = 0 (modelNo).

2.4 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.

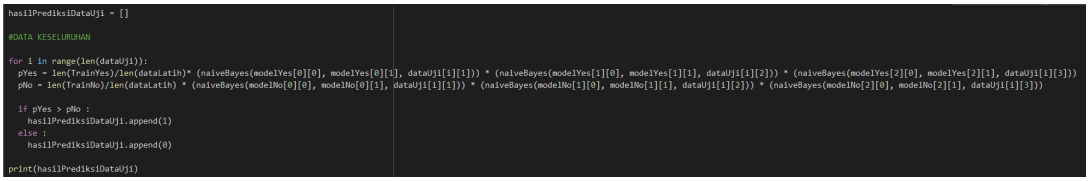
$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right)$$

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

```
[ ]  def naiveBayes(mean, variansi, number):
       a = 1 / (variansi * math.sqrt(2*3.14))
       b = math.exp(-((number - mean)**2) / (2*variansi**2))
       return a * b
```

2.5 Memprediksi Hasil Data Uji

Pada tahap ini dimulai dengan menginisiasi array yang berisi hasil prediksi data uji. Selanjutnya, akan dihitung peluang yes = 1 atau peluang no = 0 menggunakan teorema naive bayes. Hasil yang sudah ditemukan akan dimasukkan ke array hasil prediksi data uji.



```
hasilPrediksiDataUji = []

#DATA KESELURUHAN

for i in range(len(dataUji)):
  pYes = len(TrainYes)/len(dataLatih)* (naiveBayes(modelYes[0][0], modelYes[0][1], dataUji[i][1])) * (naiveBayes(modelYes[1][0], modelYes[1][1], dataUji[i][2])) * (naiveBayes(modelYes[2][0], modelYes[2][1], dataUji[i][3]))
  pNo = len(TrainNo)/len(dataLatih) * (naiveBayes(modelNo[0][0], modelNo[0][1], dataUji[i][1])) * (naiveBayes(modelNo[1][0], modelNo[1][1], dataUji[i][2])) * (naiveBayes(modelNo[2][0], modelNo[2][1], dataUji[i][3]))

  if pYes > pNo :
    hasilPrediksiDataUji.append(1)
  else :
    hasilPrediksiDataUji.append(0)

print(hasilPrediksiDataUji)
```

Berikut merupakan array hasil prediksi pada data uji.

```
---HASIL PREDIKASI DATA UJI---
[0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

2.6 Menghitung Akurasi Data Uji

Pertama - tama kita menginisiasi variabel tp (true positif) = 0, tn (true negatif) = 0, fp (false positif) = 0, fn (false negatif) = 0. Setelah itu, melakukan perulangan sebanyak jumlah data uji untuk menentukan data uji termasuk tp, tn, fp, atau fn. Lalu jika perulangan sudah selesai, dilanjutkan dengan menghitung nilai akurasi data.

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

hasil_akurasi = (tp+tn)/(tp+tn+fp+fn)
print("HASIL AKURASI DATA UJI : ", hasil_akurasi*100, "%")
```

Berdasarkan model yang sudah dibangun, diperoleh akurasi pada data uji sebesar 75,67 % .

```
HASIL AKURASI DATA UJI :  75.67567567567568 %
```

2.7 Memprediksi Hasil Data Keseluruhan

Pada tahap ini dimulai dengan menginisiasi array yang berisi hasil prediksi data keseluruhan (train). Selanjutnya, akan dihitung peluang yes = 1 atau peluang no = 0 menggunakan teorema naive bayes. Hasil yang sudah ditemukan akan dimasukkan ke array hasil prediksi data keseluruhan (train).

```
hasilPrediksiDataKeseluruhan = []

#DATA KESELURUHAN

for i in range(len(dataTrain)):
  pYesAll = len(TrainYes)/len(dataLatih)* (naiveBayes(modelYes[0][0], modelYes[0][1], dataTrain[i][1])) * (naiveBayes(modelYes[1][0], modelYes[1][1], dataTrain[i][2])) * (naiveBayes(modelYes[2][0], modelYes[2][1], dataTrain[i][3]))
  pNoAll = len(TrainNo)/len(dataLatih) * (naiveBayes(modelNo[0][0], modelNo[0][1], dataTrain[i][1])) * (naiveBayes(modelNo[1][0], modelNo[1][1], dataTrain[i][2])) * (naiveBayes(modelNo[2][0], modelNo[2][1], dataTrain[i][3]))

  if pYesAll > pNoAll :
    hasilPrediksiDataKeseluruhan.append(1)
  else :
    hasilPrediksiDataKeseluruhan.append(0)

print(hasilPrediksiDataKeseluruhan)
```

Berikut merupakan array hasil prediksi pada data keseluruhan.

```
---HASIL PREDIKSI DATA KESELURUHAN----
[1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
```

2.8 Menghitung Akurasi Data Keseluruhan

Pertama - tama kita menginisiasi variabel tp (true positif) = 0, tn (true negatif) = 0, fp (false positif) = 0, fn (false negatif) = 0. Setelah itu, melakukan perulangan sebanyak jumlah data uji untuk menentukan data uji termasuk tp, tn, fp, atau fn. Lalu jika perulangan sudah selesai, dilanjutkan dengan menghitung nilai akurasi data.

```
tp_ALL = 0
tn_ALL = 0
fp_ALL = 0
fn_ALL = 0
for i in range(len(dataTrain)):
  if hasilPrediksiDataKeseluruhan[i] == 1 and dataTrain[i][4] == 1:
    tp_ALL = tp_ALL + 1
  elif hasilPrediksiDataKeseluruhan[i] == 0 and dataTrain[i][4] == 0:
    tn_ALL = tn_ALL + 1
  elif hasilPrediksiDataKeseluruhan[i] == 1 and dataTrain[i][4] == 0:
    fp_ALL = fp_ALL + 1
  elif hasilPrediksiDataKeseluruhan[i] == 0 and dataTrain[i][4] == 1:
    fn_ALL = fn + 1

hasil_akurasi = (tp_ALL+tn_ALL)/(tp_ALL+tn_ALL+fp_ALL+fn_ALL)
print("HASIL AKURASI DATA KESELURUHAN : ", hasil_akurasi*100, "%")
```

Berdasarkan model yang sudah dibangun, diperoleh akurasi pada data keseluruhan sebesar 77,62% .

```
HASIL AKURASI DATA KESELURUHAN :  77.62237762237763 %
```

2.9 Hasil Data Testing

Berdasarkan nilai akurasi model pada data uji dan data keseluruhan, menurut kami model tersebut cukup baik untuk dipakai dalam memprediksi output pada data testing. Berikut adalah kode dalam proses menentukan output pada data testing.

```
dataTesting = np.array(df_test)

for i in range(len(dataTesting)):
  pYes = len(TrainYes)/len(dataLatih) *(naiveBayes(modelYes[0][0], modelYes[0][1], dataTesting[i][1])) * (naiveBayes(modelYes[1][0], modelYes[1][1], dataTesting[i][2])) * (naiveBayes(modelYes[2][0], modelYes[2][1], dataTesting[i][3]))
  pNo = len(TrainNo)/len(dataLatih) * (naiveBayes(modelNo[0][0], modelNo[0][1], dataTesting[i][1])) * (naiveBayes(modelNo[1][0], modelNo[1][1], dataTesting[i][2])) * (naiveBayes(modelNo[2][0], modelNo[2][1], dataTesting[i][3]))

  if pYes > pNo :
    dataTesting[i][4] = 1
  else :
    dataTesting[i][4] = 0

print("---HASIL DATA TESTING---")

Hasil_DataTesting = pd.DataFrame(dataTesting, columns =['ID', 'X1', 'X2', 'X3', 'y'])
print(Hasil_DataTesting)

#MEMBUAT FILE EXCEL
Hasil_DataTesting.to_excel('TUPRO3.xlsx')
```

Berdasarkan proses di atas, maka diperoleh hasil output pada data testing sebagai berikut.

```
---HASIL DATA TESTING---
    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
```

Kemudian output tersebut disimpan ke dalam file excel

```
#MEMBUAT FILE EXCEL
Hasil_DataTesting.to_excel('TUPRO3.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 klasifikasi data train menggunakan metode naive bayes maka dapat disimpulkan bahwa metode naive bayes dapat digunakan dalam mengklasifikasikan data dengan metode probabilitas dan statistik, yaitu berdasarkan pengalaman di masa sebelumnya dapat digunakan untuk memprediksi peluang di masa depan. Nilai persentase keakuratan menunjukkan keefektifan dataset yang diterapkan ke dalam metode Naive Bayes Classification.

Lampiran

- Pembagian Tugas

Nama	Jobdesk	Kontribusi	Link Video
Aisyah Dliya Ramadhanti	Membuat program dan laporan	50%	https://youtu.be/b1emp9hf4XQ
Fijar Yasmina Pritama	Membuat Program dan laporan	50%	https://youtu.be/-ezK6c2OWGQ

- Link Google Colab
  https://colab.research.google.com/drive/1NmvtEmg1H8IBGdEiAqSSD37zig_PESZa?usp=sharing