LAPORAN

TUGAS PEMROGRAMAN

LEARNING

Disusun untuk memenuhi tugas

Mata Kuliah: Pengantar Kecerdasan Buatan
Dosen Pengampu: MAHMUD DWI SULISTIYO, S.T., M.T., PH.D.

Oleh:

DIAN RAMADHINI (1301200254)
MUHAMMAD RAFI ANDEO PRAJA (1301200278)

KELAS IF-44-01
JURUSAN S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM

DAFTAR ISI

BAB I

PENDAHULUAN

A. Latar Belakang Masalah

Klasifikasi Naive Bayesian adalah metode klasifikasi yang paling dikenal dan digunakan. Bukan hanya itu, Naive Bayesian juga mudah diimplementasikan pada berbagai jenis kumpulan data, tetapi juga cukup efisien. Misalkan X adalah sampel data yang tidak memiliki label kelas. Biarkan H menjadi hipotesis sedemikian rupa sehingga X milik kelas tertentu. Memastikan P(H|X) probabilitas hipotesis H sesuai dengan yang diberikan X yang diamati. P(H|X) adalah posterior probabilitas yang mewakili keyakinan dalam hipotesis setelah X diberikan. Teorema Bayesian menyediakan cara untuk menghitung probabilitas posterior P(H|X) menggunakan probabilitas P(H), P(X), dan P(X|H).

B. Rumusan Masalah

Pada tahap training, output harus menghasilkan model yang sesuai dengan metode yang dipilih, yaitu Naïve Bayes. Pada tahap testing, output harus berupa kelas (0 atau 1); Lebih jauh lagi, jika ada lebih dari satu record/baris sebagai input data untuk tahap *testing*, maka program dapat mengeluarkan list output kelas yang bersesuaian dengan setiap baris data *testing* tersebut.

C. Tujuan

Mencocokkan output program untuk data uji *training* dan *testing* dengan target atau kelas sesungguhnya.

BAB II

PEMBAHASAN

A. Metode Learning

Metode learning yang kami gunakan adalah metode learning menggunakan algoritma Naive Bayes yang dimana data akan dilatih menggunakan pendekatan probabilitas. Naive Bayes sendiri merupakan suatu algoritma yang mudah untuk dipahami karena tidak hanya menggunakan pendekatan probabilitas tetapi juga menggunakan pendekatan statistika yang dimana algoritma tersebut memerlukan pencarian standar deviasi dan rata - rata dari suatu data yang ingin dilatih. Kami memilih algoritma Naive Bayes karena algoritma tersebut dapat diimplementasikan ke dalam bahasa pemrograman dengan mudah dan efisien dalam penulisan kode.

B. Pelatihan atau Training Model

Dalam algoritma Naive Bayes, pelatihan atau training model dilakukan dengan menentukan rata - rata dan standar deviasi dari suatu data training terlebih dahulu. Kedua nilai tersebut nantinya akan digunakan kembali untuk sebuah fungsi probabilitas yang bernama *"Gauss Distribution"* atau fungsi distribusi normal yang memiliki rumus sebagai berikut:

$$P(x_k|C_i) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} e^{-\frac{(x_k-\mu_{ik})^2}{2\sigma_{ik}^2}}$$

Fungsi tersebut nantinya akan menghasilkan sebuah peluang sebagai training model dari data yang akan digunakan. Penggunaan fungsi tersebut hanya diperlukan jika data yang diolah berbentuk kontinu (hanya menampilkan nilai tidak atribut), dikarenakan data yang kami miliki tidak berbentuk atribut maka kami menggunakan fungsi tersebut untuk dapat memberikan training model yang akan digunakan untuk data testing.

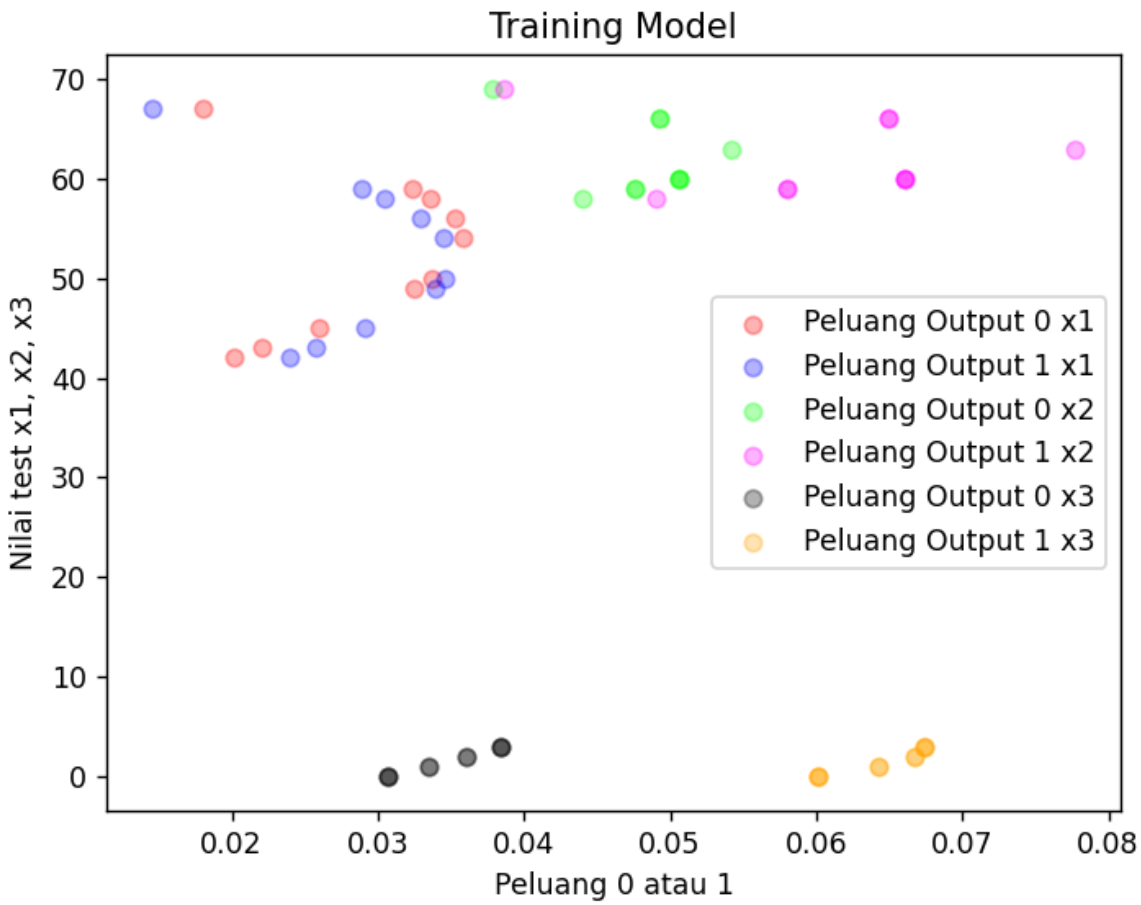
Fungsi *"Gauss Distribution"* dalam pemrograman:

```
# function normal distribution or gauss probability
def normal(x, average, stdev):
    prob = 1 / np.sqrt(2 * np.pi * stdev**2)
    return prob * np.exp(-0.5 / stdev**2 * (x - average)**2)
```

Keterangan:
X: data testing
Average: rata - rata dari data training
Stdev: standar deviasi dari data training

C. Model Hasil Training

Model hasil training yang didapatkan adalah output dari nilai fungsi *"Gauss Distribution"* atau fungsi distribusi normal tersebut yang dimana kami tampilkan berbentuk scatter plot berikut:



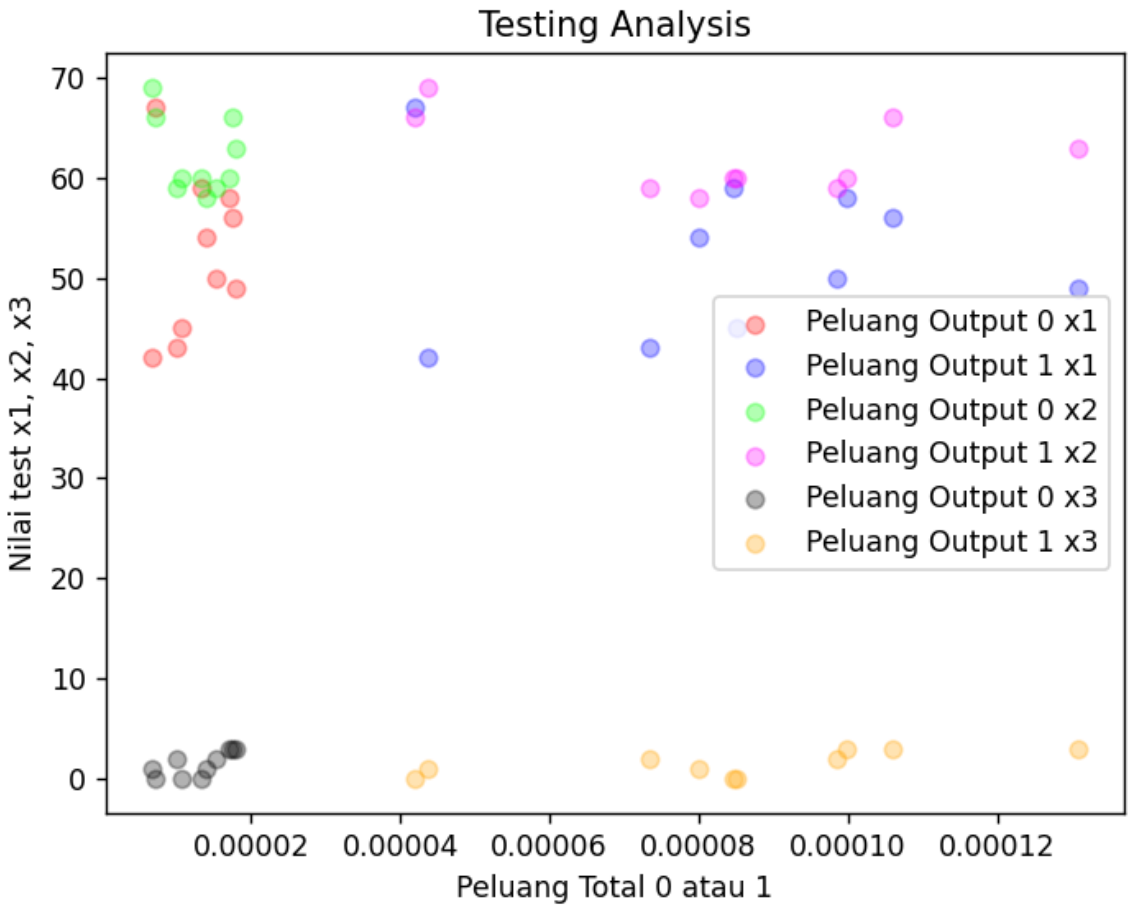
Keterangan:
Plot merepresentasikan suatu nilai x1, x2, dan x3 yang memiliki peluang output 0 lebih besar dari 1 atau peluang output 1 lebih besar dari 0. Semakin besar nilai peluang maka plot akan terletak paling kanan dan sebaliknya, semakin besar nilai test maka plot akan terletak paling atas dan sebaliknya. Warna plot juga menandakan apakah peluang output data test tersebut 0 atau 1.

D. Pengujian atau Testing Model

Setelah mendapatkan output dari fungsi *"Gauss Distribution"* atau fungsi distribusi normal, hasil peluang tersebut akan dikalikan dengan semua nilai peluang dari masing - masing data yang dimana untuk peluang output 0 memiliki jumlah data sebanyak 78, sedangkan untuk peluang output 1 memiliki jumlah data 218 yang dimana total data training adalah 296 data. Rumus dari peluang total tersebut adalah:

Ptotal 0 = 78/296 * P(Xk|Ci)
Ptotal 1 = 218/296 * P(Xk|Ci)

Kami mendapatkan hasil dari peluang total tersebut yang dimana seluruh peluang data testing output 0 lebih kecil daripada seluruh peluang data testing output 1. Berikut adalah output peluang dalam scatter plot:



Keterangan:
Plot merepresentasikan peluang hasil dari gabungan nilai test x1, x2, dan x3. Dapat dicermati bahwa seluruh nilai Peluang Total Output 1 lebih besar dari seluruh nilai Peluang Total Output 0, dapat disimpulkan bahwa gabungan nilai test x1, x2, dan x3 semuanya mendapatkan output bernilai 1.

Fungsi dalam pemrograman:

```
# function accumulated probability for output 0
def peluang_total_0(peluang_0_x1, peluang_0_x2, peluang_0_x3):
    return 78/296 * peluang_0_x1 * peluang_0_x2 * peluang_0_x3

# function accumulated probability for output 1
def peluang_total_1(peluang_1_x1, peluang_1_x2, peluang_1_x3):
    return 218/296 * peluang_1_x1 * peluang_1_x2 * peluang_1_x3
```

E. Evaluasi Model

Kami menggunakan fungsi akurasi untuk menentukan apakah hasil dari training dan testing sesuai atau tidak. Fungsi akurasi sendiri memiliki rumus sederhana sebagai berikut:

$$ACC = \frac{T}{P}$$

Keterangan:
T = True Values
P = Prediction Values

Fungsi dalam pemrograman:

```
# function accuracy
def akurasi(prediction):
    true_values = np.array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])
    prediction_values = np.array([[prediction]])

    n = true_values.shape[1]
    accuracy = (true_values == prediction_values).sum() / n

    return accuracy
```

BAB III

PENUTUPAN

A. Kesimpulan

Dari program yang telah kami buat, dapat disimpulkan bahwa dalam algoritma Naive Bayes, training model dilakukan dengan menentukan rata-rata dan standar deviasi dari suatu data training terlebih dahulu, kedua nilai tersebut nanti akan digunakan kembali untuk fungsi probabilitas *Gauss Distribution*. Pada tahap testing, output dari fungsi *Gauss Distribution* tersebut akan dikalikan dengan semua nilai peluang dari masing-masing data. Jadi, pada tahap training dan testing jika semakin besar nilai peluang yang didapat maka plot akan terletak paling kanan dan sebaliknya. Jika kita ingin mengetahui hasil dari training dan testing sesuai atau tidak, maka kita dapat menggunakan fungsi akurasi untuk melakukan pengecekan dengan rumus ACC = T/P.

B. Saran

Kami menyadari bahwa laporan dan program kami masih jauh dari kata sempurna, maka dari itu kami berharap untuk kedepannya dapat lebih baik dan lebih teliti dalam tugas pada mata kuliah berikutnya. Semoga laporan dan program kami dapat memberi manfaat bagi siapapun yang tertarik mengenai naïve bayes. Kami mengucapkan terima kasih banyak kepada bapak MAHMUD DWI SULISTIYO, S.T., M.T., PH.D. sebagai dosen pengampu mata kuliah pengantar kecerdasan buatan yang telah memberikan kami kesempatan sebagai mahasiswa untuk belajar dan mencari ilmu sebanyak-banyaknya mengenai naïve bayes ini. Kami berharap bisa mendapatkan kritik dan saran dari siapapun yang membaca laporan dan mencoba program kami, kami menerima segala kritik dan saran yang dapat memperbaiki hasil kinerja kami.

C. Peran Anggota Kelompok

Dian Ramadhini : Slide dan Laporan
Muhammad Rafi Andeo Praja : Program dan Laporan pada bagian pembahasan.

DAFTAR PUSTAKA

Cagatay. Catal, Ugur. Sevim, Banu. Diri. (2010). Practical Development of an Eclipse-based Software Fault Prediction Tool Using Naive Bayes Algorithm. Diambil 15 Juni 2022.

Kevin. P. Murphy. (2007). Conjugate Bayesian Analysis of the Gaussian Distribution. Diambil 18 Juni 2022.

LAMPIRAN

Program:

```
"""
Python Program for Naive Bayes Learning
Kelompok 2:
DIAN RAMADHINI (1301200254)
MUHAMMAD RAFI ANDEO PRAJA (1301200278)
"""

# import library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# read excel data
xls = pd.ExcelFile('traintest.xlsx')
df_train = pd.read_excel(xls, 'train')
df_train_1 = pd.read_excel(xls, 'train y = 1')
df_train_0 = pd.read_excel(xls, 'train y = 0')
df_test = pd.read_excel(xls, 'test')

# drop id
df_train_1 = df_train_1.drop('id', axis=1)
df_train_0 = df_train_0.drop('id', axis=1)
df_test = df_test.drop('id', axis=1)

# new variable for each x with output 0
df_train_0_x1 = df_train_0['x1']
df_train_0_x2 = df_train_0['x2']
df_train_0_x3 = df_train_0['x3']

# new variable for each x with output 1
df_train_1_x1 = df_train_1['x1']
df_train_1_x2 = df_train_1['x2']
df_train_1_x3 = df_train_1['x3']

# new variable for each x in test section
df_test_x1 = df_test['x1']
```

```
df_test_x2 = df_test['x2']
df_test_x3 = df_test['x3']

# function average for output 0
def average_0(arr):
    i = 0
    total = 0
    while i < 78:
        total = total + arr[i]
        i += 1
    rata = total / i
    return rata

# function average for output 1
def average_1(arr):
    i = 0
    total = 0
    while i < 218:
        total = total + arr[i]
        i += 1
    rata = total / i
    return rata

# assign variable with average function for output 0
rata_x1_0 = average_0(df_train_0_x1)
rata_x2_0 = average_0(df_train_0_x2)
rata_x3_0 = average_0(df_train_0_x3)

# assign variable with average function for output 1
rata_x1_1 = average_1(df_train_1_x1)
rata_x2_1 = average_1(df_train_1_x2)
rata_x3_1 = average_1(df_train_1_x3)

# new variable for stdev output 0
stdev_x1_0 = np.std(df_train_0_x1)
stdev_x2_0 = np.std(df_train_0_x2)
stdev_x3_0 = np.std(df_train_0_x3)

# new variable for stdev output 1
stdev_x1_1 = np.std(df_train_1_x1)
```

```
stdev_x2_1 = np.std(df_train_1_x2)
stdev_x3_1 = np.std(df_train_1_x3)

# function normal distribution or gauss probability
def normal(x, average, stdev):
    prob = 1 / np.sqrt(2 * np.pi * stdev**2)
    return prob * np.exp(-0.5 / stdev**2 * (x - average)**2)

# function accumulated probability for output 0
def peluang_total_0(peluang_0_x1, peluang_0_x2, peluang_0_x3):
    return 78/296 * peluang_0_x1 * peluang_0_x2 * peluang_0_x3

# function accumulated probability for output 1
def peluang_total_1(peluang_1_x1, peluang_1_x2, peluang_1_x3):
    return 218/296 * peluang_1_x1 * peluang_1_x2 * peluang_1_x3

# new variable with normal function for output 0
peluang_0_x1 = normal(df_test_x1, rata_x1_0, stdev_x1_0)
peluang_0_x2 = normal(df_test_x2, rata_x2_0, stdev_x2_0)
peluang_0_x3 = normal(df_test_x3, rata_x3_0, stdev_x3_0)

# new variable with normal function for output 1
peluang_1_x1 = normal(df_test_x1, rata_x1_1, stdev_x1_1)
peluang_1_x2 = normal(df_test_x2, rata_x2_1, stdev_x2_1)
peluang_1_x3 = normal(df_test_x3, rata_x3_1, stdev_x3_1)

# new variable for accumulated all probability
peluang_0_total = peluang_total_0(peluang_0_x1, peluang_0_x2,
peluang_0_x3)
peluang_1_total = peluang_total_1(peluang_1_x1, peluang_1_x2,
peluang_1_x3)

# function prediction
def prediksi(peluang_0_total, peluang_1_total):
    hasil = 0
    if peluang_0_total > peluang_1_total:
        hasil = 0
    elif peluang_1_total > peluang_0_total:
        hasil = 1
    return hasil
```

```
# training model
plt.title("Training Model")
plt.xlabel("Peluang 0 atau 1")
plt.ylabel("Nilai test x1, x2, x3")
plt.scatter(peluang_0_x1, df_test_x1, color = "red", label = "Peluang
Output 0 x1", alpha = 0.3)
plt.scatter(peluang_1_x1, df_test_x1, color = "blue", label = "Peluang
Output 1 x1", alpha = 0.3)
plt.scatter(peluang_0_x2, df_test_x2, color = "lime", label = "Peluang
Output 0 x2", alpha = 0.3)
plt.scatter(peluang_1_x2, df_test_x2, color = "magenta", label = "Peluang
Output 1 x2", alpha = 0.3)
plt.scatter(peluang_0_x3, df_test_x3, color = "black", label = "Peluang
Output 0 x3", alpha = 0.3)
plt.scatter(peluang_1_x3, df_test_x3, color = "orange", label = "Peluang
Output 1 x3", alpha = 0.3)
plt.legend()
plt.show()

plt.title("Testing Analysis")
plt.xlabel("Peluang Total 0 atau 1")
plt.ylabel("Nilai test x1, x2, x3")
plt.scatter(peluang_0_total, df_test_x1, color = "red", label = "Peluang
Output 0 x1", alpha = 0.3)
plt.scatter(peluang_1_total, df_test_x1, color = "blue", label = "Peluang
Output 1 x1", alpha = 0.3)
plt.scatter(peluang_0_total, df_test_x2, color = "lime", label = "Peluang
Output 0 x2", alpha = 0.3)
plt.scatter(peluang_1_total, df_test_x2, color = "magenta", label =
"Peluang Output 1 x2", alpha = 0.3)
plt.scatter(peluang_0_total, df_test_x3, color = "black", label = "Peluang
Output 0 x3", alpha = 0.3)
plt.scatter(peluang_1_total, df_test_x3, color = "orange", label =
"Peluang Output 1 x3", alpha = 0.3)
plt.legend()
plt.show()

# function accuracy
def akurasi(prediction):
```

```
    true_values = np.array([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])
    prediction_values = np.array([[prediction]])

    n = true_values.shape[1]
    accuracy = (true_values == prediction_values).sum() / n

    return accuracy

# output to file
i = 0
prediction = [None] * 10
while i < 10:
  prediction[i] = prediksi(peluang_0_total[i], peluang_1_total[i])
  i += 1

print(akurasi(prediction))

Result = pd.DataFrame(prediction, columns=['Testing Output'])
Result.to_excel('Result.xlsx')
```