Praktikum 4
Algoritma Klasifikasi Naive Bayes

Tujuan

Setelah mengikuti praktikum ini, mahasiswa diharapkan mampu:

1. Mengetahui variasi algoritma Naive Bayes
2. Mampu mengimplementasikan algoritma Naive Bayes Gaussian
3. Mampu mengimplementasikan algoritma Naive Bayes Multinomial

Dasar Teori

Klasifikasi Naive Bayes merupakan sebuah metode klasifikasi sederhana yang berdasarkan teori peluang bernama Teorema Bayes. Peluang sebuah data *d* memiliki kelas *c* dapat dihitung dengan persamaan berikut:

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)}$$

Keterangan:

- $P(c|d)$: posterior, yaitu peluang kelas *c* diberikan data *d*
- $P(c)$: prior, yaitu peluang awal munculnya kelas *c*
- $P(d|c)$: likelihood atau conditional probability
- $P(d)$: evidence, yaitu peluang munculnya data *d*

Kelas dari suatu data uji ditentukan berdasarkan nilai posterior terbesar, atau secara matematis dapat ditulis sebagai:

$$c(d) = arg\ max\ P(c|d)$$

Pada algoritma naive bayes, nilai evidence selalu sama untuk setiap kelas sehingga dapat diabaikan. Dengan demikian, persamaan posterior pada algoritma Naive Bayes dapat diubah menjadi:

$$P(c|d) = P(c) \prod_{w \epsilon d} P(w|c)$$

Dimana $\prod_{w \epsilon d} P(w|c)$ merupakan perkalian dari conditional probability masing-masing fitur yang terdapat pada data *d*.

Nilai likelihood atau conditional probability dihitung menggunakan salah satu model yang ada, berdasarkan jenis datanya:

- Bernoulli model, digunakan jika fitur *w* memiliki data bertipe biner

$$P(w|c) = b_t \frac{N_c(w)}{N_c} + (1 - b_t)\,(1 - \frac{N_c(w)}{N_c})$$

  Jika fitur *w* ada pada data berkategori *c*, $b_t = 1$
  Jika fitur *w* tidak ada pada data berkategori *c*, $b_t = 0$
  $Nc(w)$ = banyaknya data berkategori *c* yang mengandung fitur *w*
  Nc = banyaknya data berkategori *c*

- Multinomial model, digunakan ketika fitur *w* memiliki data diskrit atau kategori

$$P(w|c) = \frac{count(w,c)}{count(c)}$$

  $Nc(w)$ = banyaknya data berkategori *c* yang mengandung fitur *w*
  Nc = banyaknya data berkategori *c*

- Gaussian model, digunakan jika fitur *w* memiliki data bertipe kontinu

$$P(w|c) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{(\frac{-(w-\mu_c)^2}{2{\sigma_c}^2})}$$

Tahapan training pada algoritma Naive Bayes adalah :
1. Menghitung prior masing-masing kategori
2. Menghitung conditional probability per fitur per kategori
3. Outputnya adalah model (nilai prior dan conditional probability)

Tahapan testing pada algoritma Naive Bayes adalah:
1. Menghitung posterior untuk masing-masing kelas.
2. Menentukan kelas data latih berdasarkan nilai posterior terbesar.

Praktikum

1. Import Data

Unduh dataset yang akan digunakan pada praktikum kali ini. Anda dapat menggunakan aplikasi wget untuk mendowload dataset dan menyimpannya dalam Google Colab. Jalankan cell di bawah ini untuk mengunduh dataset

```
! wget
https://gist.githubusercontent.com/netj/8836201/raw/6f9306ad21398
ea43cba4f7d537619d0e07d5ae3/iris.csv
```

Setelah dataset berhasil diunduh, langkah berikutnya adalah membaca dataset dengan memanfaatkan fungsi readcsv dari library pandas. Lakukan pembacaan berkas csv ke dalam dataframe dengan nama data menggunakan fungsi readcsv. Jangan lupa untuk melakukan import library pandas terlebih dahulu

```
import pandas as pd
import numpy as np
data = pd.read_csv('iris.csv')
```

Cek isi dataset Anda dengan menggunakan perintah **head()**

```
data.head()
```

2. Membagi Data Menjadi Data Latih dan Data Uji

Metode pembelajaran mesin memerlukan dua jenis data :

1. Data latih : Digunakan untuk proses training metode klasifikasi
2. Data uji : Digunakan untuk proses evaluasi metode klasifikasi

Data uji dan data latih perlu dibuat terpisah (mutualy exclusive) agar hasil evaluasi lebih akurat. Data uji dan data latih dapat dibuat dengan cara membagi dataset dengan rasio tertentu, misalnya 80% data latih dan 20% data uji.

Library Scikit-learn memiliki fungsi train_test_split pada modul model_selection untuk membagi dataset menjadi data latih dan data uji. Bagilah dataset anda menjadi dua, yaitu **data_latih** dan **data_uji**.

```
from sklearn.model_selection import train_test_split
data_latih,data_uji = train_test_split(data,test_size=0.2)
```

Parameter **test_size** pada fungsi **train_test_split** menunjukkan persentase data uji yang dihasilkan. Pada contoh ini, 20% data digunakan sebagai data uji.

Tampilkan banyaknya data pada **data_latih** dan **data_uji**. Seharusnya **data_latih** terdiri dari 120 data, dan **data_uji** terdiri dari 30 data

```
print(data_uji.shape[0])
print(data_latih.shape[0])
```

Pisahkan label/kategori dari **data uji** menjadi variabel tersendiri. Beri nama variabelnya **label_uji** Fungsi **pop()** pada dataframe berfungsi untuk mengambil suatu kolom dari dataframe, hampir sama dengan fungsi pop() pada struktur data stack.

```
label_uji = data_uji.pop('variety')
```

Cek data latih dan data uji setelah melalui operasi pop().

```
data_uji.head()
```

3. Menghitung Prior

Tahapan pertama pada algoritma Naive Bayes adalah perhitungan prior. Prior suatu kelas merupakan peluang awal munculnya kelas tersebut. Atau dengan kata lain, prior merupakan frekuensi relatif dari suatu kelas terhadap keseluruhan data.

Buatlah fungsi bernama **hitung_prior** yang berfungsi menghitung nilai prior dari masing-masing kelas yang terdapat pada data latih. Fungsi ini menerima input berupa list kelas dari data latih.

```
from collections import Counter
def hitung_prior(list_kelas):
  n_data = len(list_kelas)
  prior = Counter(list_kelas)
  for key in prior.keys():
    prior[key]=prior[key]/n_data
  return prior
```

Fungsi **hitung_prior** memanfaatkan fungsi Counter dari modul collections yang berfungsi menghitung frekuensi masing-masing kelas dari data latih. Selanjutnya, frekuensi masing-masing kelas dibagi dengan banyaknya data untuk mendapatkan nilai prior

Ujilah fungsi **hitung_prior** menggunakan kelas dari variabel **data_latih**. Amati hasilnya

```
prior = hitung_prior(data_latih['variety'])
print(prior)
```

4. Menghitung likelihood

Praktikum kali ini mengimplementasikan metode Naive Bayes menggunakan datset iris. Semua fitur pada dataset iris memiliki tipe numerik. Dengan demikian, perhitungan likelihood dilakukan menggunakan Gaussian Model.

Tahapan yang perlu dilakukan sebelum perhitungan likelihood adalah menghitung rata-rata dan deviasi standar masing-masing fitur per kelas. Buatlah fungsi **hitung_rata2_std_kelas** yang berfungsi menghitung rata-rata dan standar deviasi per fitur dan kelas

```
def hitung_rata2_std_kelas(input_data):
  list_columns = input_data.columns[:-1]
  class_column_name = input_data.columns[-1]
  list_class = set(input_data[class_column_name])
  rata2 = {}
  std = {}
  for column in list_columns:
    for a_class in list_class:
        rata2[(a_class,column)] =
input_data.loc[input_data[class_column_name]==a_class][column].mean()
        std[(a_class,column)] =
input_data.loc[input_data[class_column_name]==a_class][column].std()
  return(rata2,std)
```

Variabel **list_columns** berisi nama-nama kolom pada dataframe, kecuali kolom terakhir yang berupa kelas data sedangkan variabel **class_column_name** berisi nama kolom dari kelas data, yaitu kolom terakhir. Variabel **list_class** berisi himpunan nama-nama kelas dari data secara unik. Variabel **rata2** dan **std** berisi rata-rata dan standar deviasi dari setiap fitur(kolom) data. Kedua variabel tersebut berupa dictionary dengan key berupa tuple(nama kelas, nama kolom).

Buatlah fungsi **hitung_likelihood_gaussian** yang bertujuan menghitung nilai likelihood suatu nilai terhadap rata-rata dan deviasi standar tertentu.

```
import math
def hitung_likelihood_gaussian(data,rata2,std):
  hasil = (1/math.sqrt(2*math.pi*(std**2)))*math.exp((-1*((data-
rata2)**2))/(2*(std**2)))
  return hasil
```

Perhitungan likelihood menggunakan model Gaussian karena data yang digunakan merupakan data kontinu. Hati-hati ketika menuliskan kode tersebut, terutama penulisan tanda kurung.

5. Proses Training

Proses training pada Naive Bayes dengan model Gaussian dilakukan untuk menghitung prior serta rata-rata dan deviasi standar pada masing-msaing fitur dan kelas.

```
def training_naive_bayes_gaussian(data_latih):
  class_column_name = data_latih.columns[-1]
  prior = hitung_prior(data_latih[class_column_name])
  (rata2,std) = hitung_rata2_std_kelas(data_latih)
  list_class = set(data_latih[class_column_name])
  list_columns = data_latih.columns[:-1]
  model = {}
  model['prior'] = prior
  model['rata2'] = rata2
```

```
    model['std'] = std
    model['list_class'] = list_class
    model['list_columns'] = list_columns
    return model
```

Lakukan proses training dan simpan hasil training pada variabel **model**

```
model = training_naive_bayes_gaussian(data_latih)
print(model)
```

Nilai rata-rata dan standar deviasi per fitur diperoleh dari fungsi **hitung_rata2_std_kelas** sedangkan nilai prior diperoleh dari fungsi **hitung_prior**. Proses training menghasilkan model yang berisi nilai-nilai berikut:

a. Nilai prior
b. Nilai rata-rata per fitur
c. Nilai standar deviasi per fitur
d. Daftar kelas pada data
e. Daftar kolom pada data

6. Proses Testing

Proses testing dilakukan dengan menghitung nilai posterior dari data uji berdasarkan nilai model yang diperoleh saat training. Nilai posterior diperoleh dari perkalian prior dengan *likelihood* masing-masing fiturnya. Setelah itu, penentuan kelas data uji dilakukan berdasarkan nilai posterior terbesar.

```
def testing_naive_bayes_gaussian(model,data_uji):
  prior = model['prior']
  rata2 = model['rata2']
  std = model['std']
  list_class = model['list_class']
  list_columns = model['list_columns']
  posterior = dict.fromkeys(list_class,1)
  for a_class in list_class:
    for column in list_columns:
      posterior[a_class] =
posterior[a_class]*hitung_likelihood_gaussian(data_uji[column],rat
a2[(a_class,column)],std[(a_class,column)])
    posterior[a_class] = posterior[a_class] * prior[a_class]
  kelas_uji = max(posterior,key=posterior.get)
  return kelas_uji
```

Pada proses testing, nilai prior serta rata-rata dan standar deviasi dari masing-masing fitur diperoleh dari model hasil training. Selanjutnya, dilakukan looping per kolom data uji untuk menghitung nilai likelihood pada fitur tersebut. Penghitungan likelihood dilakukan

dengan memanggil fungsi **hitung_likelihood_gaussian** dengan parameter input berupa nilai data uji pada kolom tersebut, rata-rata kolom, dan standar deviasi kolom. Nilai likelihood pada setiap kolom dikalikan untuk mendapatkan nilai likelihood total. Setelah itu, nilai posterior kelas diperoleh dari perkalian likelihood total dengan nilai prior kelas tersebut. Kelas data uji ditentukan berdasarkan nilai maksimal posterior

Ujilah fungsi **testing_naive_bayes_gaussian** menggunakan data uji pertama. Bandingkan hasilnya dengan nilai label sebenarnya.

```
indeks_uji = 0
prediksi =
testing_naive_bayes_gaussian(model,data_uji.iloc[indeks_uji])
print(prediksi)
print(label_uji.iloc[indeks_uji])
```

Lakukan pengujian untuk semua data uji. Selanjutnya bandingkan dengan label sebenarnya untuk menghitung berapa banyak data uji yang berhasil diprediksi dengan benar

```
prediksi_total =[]
for indeks in range(data_uji.shape[0]):
      prediksi_total.append(testing_naive_bayes_gaussian(model,dat
a_uji.iloc[indeks]))
print("Total prediksi benar: ",sum(prediksi_total==label_uji))
```

Tugas

Pada tugas kali ini Anda akan menggunakan dataset Car Evaluation Dataset yang telah dimodifikasi. Fitur-fitur yang ada bertipe kategori sebagai berikut:

- **buying**. Merupakan harga beli mobil. Nilai : vhigh, high, med, low
- **maint**. Menandakan biaya perawatan mobil. Nilai: vhigh, high, med, low
- **lug_boot**. Menandakan ukuran bagasi. Nilai: small, med, big
- **safety**. Menandakan skor keamanan mobil. Nilai: low, med, high
- **class**. Merupakan kelas data. Nilai: unacc, acc, good, vgood

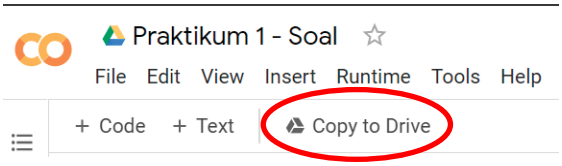
1. Download dataset dari Google Classroom dan simpan ke dalam variabel **data_tugas**.
2. Bagi data menjadi data latih dan uji dengan nama **tugas_latih** dan **tugas_uji** dengan rasio 70%:30%
3. Pisahkan label/kategori dari data uji menjadi variabel tersendiri. Beri nama variabelnya **tugas_label_uji**
4. Hitunglah nilai prior dari data latih dan simpan nilainya pada variabel bernama **prior_tugas**. Anda dapat menggunakan fungsi **hitung_prior** yang telah Anda buat.
5. Buatlah fungsi bernama **hitung_likelihood_multinomial** dengan parameter input

berupa data latih. Fungsi ini bertujuan menghitung nilai likelihood pada masing-masing kolom pada data latih.

6. Buatlah fungsi bernama **training_naive_bayes_multinomial** dengan input berupa data latih.
7. Buatlah fungsi **testing_naive_bayes_multinomial** untuk mendapatkan kelas dari sebuah data uji.
8. Lakukan pengujian, baik terhadap sebuah data uji maupun semua data uji.

Petunjuk pengerjaan soal:

1. Klik link berikut, pastikan Anda login menggunakan **akun student UB**. https://colab.research.google.com/drive/1JVClT6V-UC3xfhcrQxnjtJtSTlbOL4eH?usp=sharing
2. Klik tombol Copy to Drive



3. Beri nama file BAB 4-Nama-NIM.ipynb
4. Isilah cell yang kosong
5. Download file *.ipynb dengan cara klik **File -> Download .ipynb**
6. Kumpulkan file *.ipynb ke asisten