Pengantar Machine Learning

Phie Chyan, Sitti Arni, Mohammad Sofyan S. Thayf,
Immanuela Puspasari S, Anis Fitri Nur Masruriyah,
Amril Mutoi Siregar, Sukenada Andisana, Rifka Widyastuti,
Tubagus M Akhriza, Khoerul Anwar, Prastyadi Wibawa Rahayu,
Vitri Aprilla Handayani, Nurul Mukhlisah Abdal

Daftar Isi

Kata Pengantar i
Daftar Isi iii

BAB 1 PENGANTAR MACHINE LEARNING 1
Pendahuluan 1
Jenis Machine Learning 3
Bagaimana Machine Learning Bekerja? 7
Implementasi Machine Learning 8

BAB 2 TEORI MACHINE LEARNING 13
Pendahuluan 13
Data Training 15
Algoritma Machine Learning 17
Teknik Belajar Machine Learning 21

BAB 3 MACHINE LEARNING LIFECYCLE 25
Pendahuluan 25
Tahapan dalam Machine Learning 27
Tahap Persyaratan Model 27

Tahap Pemrosesan Data 28
Tahap Perancangan Fitur dan Pelatihan Model 30

BAB 4 ALGORITMA MACHINE LEARNING 35
Pendahuluan 35
Cara Kerja Machine Learning 36
Algoritma Machine Learning 43
Penerapan Machine Learning 47

BAB 5 SUPERVISED LEARNING 49
Pendahuluan 49
Penggunaan Supervised Learning 50
Tahapan Proses Supervised Learning 52
Jenis Algoritma pada Supervised Learning 55

BAB 6 REGRESI LOGISTIK 59
Pendahuluan 59
Asumsi Regresi Logistik 62
Model Persamaan Regresi Logistik 63
Jenis Regresi Logistik 64
Kelebihan Regresi Logistik 66
Kekurangan Regresi Logistik 67
Penerapan Logistic Regression 68

BAB 7 SUPPORT VECTOR MACHINES (SVM) 87
Pendahuluan 87
Pattern Recognition 90
Classification and Regression SVM 92
Soft-Margin Support Vector Machines (SVM) 93
The Non-Linier Classifier dan Kernel Trick 95
Karakteristik SVM 98
Kelebihan dan Kekurangan SVM 98

BAB 8 NAÏVE BAYES 101
Pendahuluan 101
Konsep Dasar Naïve Bayes 102
Komputasi Prediksi dalam NBC 104
Langkah Memprediksi Menggunakan NBC 106
Tipe Data yang Bisa Diproses oleh NBC 108
Implementasi 109
Matriks Evaluasi 118

BAB 9 K-NEAREST NEIGHBORS 131
Pendahuluan 131
Algoritma K-NN 133
Pemilihan K 133
Algoritma k-NN 135
Menghitung Jarak 137
Keunggulan k-NN 139
Kekurangan k-NN 139
Aplikasi k-NN dalam Machine Learning 140
Contoh Penerapan k-NN (phyton) 142

BAB 10 K-MEANS 149
Pendahuluan 149
Klasterisasi 150
K-Means 152
Langkah-Langkah Perhitungan 153
Contoh Kasus Algoritma K-Means 154

BAB 11 K-MEDOIDS CLUSTERING 159
Pendahuluan 159
Konsep Dasar K-Medoid Clustering 162
Pemilihan Jumlah Kluster 162
Implementasi K-Medoid Clustering 164
Segmentasi Pelanggan 166

BAB 12 JARINGAN SARAF TIRUAN 169
Pendahuluan 169
Pentingnya Jaringan Saraf Tiruan 170
Sejarah Jaringan Saraf Tiruan 171
Komponen Dasar Jaringan Saraf Tiruan 172
Pelatihan Jaringan Saraf Tiruan 173
Penerapan Jaringan Saraf Tiruan 176

Daftar Pustaka 179
Tentang Penulis 193

BAB 9 K-NEAREST NEIGHBORS

Pendahuluan

k-Nearest Neighbors (k-NN) adalah salah satu algoritma klasifikasi paling dasar namun penting dalam Machine Learning. Itu milik domain pembelajaran yang diawasi dan menemukan aplikasi intensif dalam pengenalan pola, penambangan data, dan deteksi intrusi. k-NN memiliki prinsip kerja dengan menemukan sekelompok *k* objek di kumpulan data pelatihan yang paling dekat dengan objek uji, dan mendasarkan pemberian label pada mayoritas kelas tertentu di lingkungan data. Hal ini mengatasi masalah bahwa dalam banyak kumpulan data, kecil kemungkinannya bahwa satu objek akan sama persis dengan objek lainnya, serta fakta bahwa informasi yang bertentangan tentang kelas suatu objek mungkin disediakan oleh objek yang paling

dekat dengannya. Ada beberapa elemen kunci dari pendekatan ini: (i) kumpulan objek berlabel yang akan digunakan untuk mengevaluasi kelas objek uji, (ii) metrik jarak atau kesamaan yang dapat digunakan untuk menghitung kedekatan objek, (iii) nilai *k*, jumlah tetangga terdekat (mayoritas), dan (iv) metode yang digunakan untuk menentukan kelas objek sasaran berdasarkan kelas dan jarak *k* tetangga terdekat. Dalam bentuknya yang paling sederhana, k-NN dapat melibatkan penetapan suatu objek kelas tetangga terdekatnya atau mayoritas tetangga terdekatnya, namun berbagai penyempurnaan dimungkinkan.

Secara lebih umum, k-NN adalah kasus khusus pembelajaran berbasis instance. Ini termasuk penalaran berbasis kasus, yang berhubungan dengan data simbolik. Pendekatan k-NN juga merupakan contoh teknik pembelajaran yang malas, yaitu teknik yang menunggu hingga kueri data terpenuhi untuk menggeneralisasi di luar data pelatihan. Meskipun klasifikasi k-NN adalah teknik klasifikasi yang mudah dipahami dan diterapkan, begitu juga klasifikasi ini berfungsi dengan baik dalam banyak situasi. Secara khusus, hasil yang diketahui dalam beberapa penelitian menunjukkan bahwa kesalahan klasifikasi oleh aturan tetangga terdekat dibatasi dua kali kesalahan berdasarkan asumsi-asumsi tertentu yang masuk akal.

Selain itu, karena kesederhanaannya, k-NN mudah dimodifikasi untuk masalah klasifikasi yang lebih rumit. Misalnya, k-NN sangat cocok untuk kelas multimodal serta aplikasi di mana suatu objek dapat memiliki banyak label kelas. Untuk mengilustrasikan poin terakhir, untuk penetapan fungsi pada gen berdasarkan profil ekspresi *microarray*, beberapa peneliti menemukan bahwa k-NN mengungguli pendekatan *support vector machine* (SVM), yang merupakan skema klasifikasi yang jauh lebih canggih.

Algoritma K-NN

Meskipun k-NN adalah aproksimator fungsi universal dalam kondisi tertentu, konsep dasarnya relatif sederhana. k-NN adalah algoritma untuk pembelajaran terawasi (*supervised*) yang hanya menyimpan contoh pelatihan berlabel.

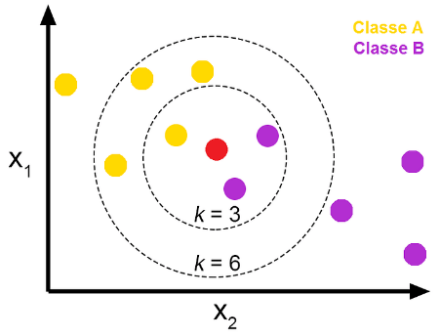
$$\left(x^{[i]}, y^{[i]}\right) \in D = (|D| = n)$$

Algoritma pembelajaran yang malas adalah pemrosesan contoh pelatihan ditunda hingga membuat prediksi lagi, pelatihan tersebut secara harfiah hanya terdiri dari penyimpanan data pelatihan. Kemudian, untuk membuat prediksi (label kelas atau target kontinu), algoritma k-NN mencari *k* tetangga terdekat dari titik kueri dan menghitung label kelas (klasifikasi) atau target kontinu (regresi) berdasarkan *k* terdekat (paling "mirip") poin. Namun, gagasan keseluruhannya adalah bahwa alih-alih memperkirakan fungsi target *f(x) = y* secara global, selama setiap prediksi, k-NN memperkirakan fungsi target secara lokal. Dalam praktiknya, lebih mudah untuk mempelajari perkiraan suatu fungsi secara lokal daripada secara global.

Pemilihan K

Pemilihan *k* atau berapa jumlahnya yang tepat menjadi penting untuk diperhatikan. Tidak ada metode yang berlaku secara umum yang bisa digunakan oleh semua orang dalam menentukan nilai ideal untuk "*k*", jadi kita harus bereksperimen dengan berbagai pilihan untuk menemukan nilai yang tepat. Untuk memilih *k* yang tepat terhadap data yang tersedia yang dapat dilakukan dengan menjalankan algoritma k-NN beberapa kali dengan nilai *k* yang berbeda

dan memilih *k* yang mengurangi jumlah kesalahan yang ditemukan dengan tetap menjaga kemampuan algoritma untuk membuat prediksi secara akurat ketika diberikan data yang belum pernah digunakan dalam pelatihan sebelumnya.



Gambar 10.1 Diagram klasifikasi k-NN

Gambar 10.1 memperlihatkan diagram perbedaan hasil klasifikasi algoritma k-NN dari penggunaan jumlah *k* yang berbeda. Ada dua kelas yaitu A dan B, kelas A diberi warna kuning dan kelas B diberi warna ungu. Sementara itu ada data lain di antara dua kelas tersebut yang berwarna merah. Tampak pada gambar 10.1, perbedaan jumlah k yang dipilih mempengaruhi hasil klasifikasi terhadap data baru. Pada saat k = 3 terdapat 1 data di kelas A dan terdapat 2 data dikelas B. Berdasarkan jumlah mayoritas kelas terdekat maka data baru terdeteksi sebagai kelas B. Namun pada saat jumlah k dinaikkan menjadi enam (k = 6) maka komposisi jumlah data menjadi berubah, yaitu kelas A sejumlah 4 data dan kelas B sejumlah 2 data. Maka, berdasarkan jumlah mayoritas kelas terdekat maka data baru dikategorikan sebagai kelas A. Oleh karena itu pemilihan jumlah k dalam algoritma k-NN mempunyai dampak yang cukup berarti terhadap hasil klasifikasi. Berikut beberapa hal yang perlu diingat.

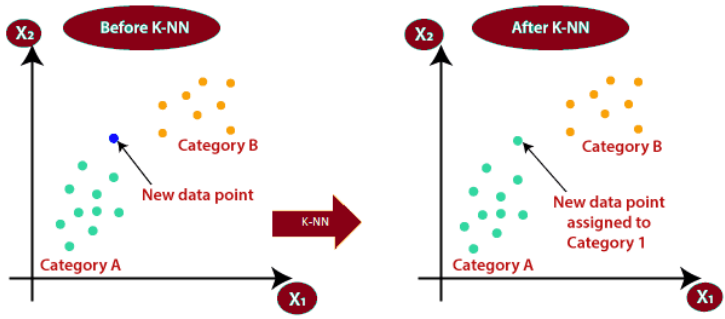
1. Saat kita memilih nilai k menjadi 1, prediksi kita

menjadi kurang stabil. Coba pikirkan sejenak, bayangkan *k=1* dan kita memiliki titik kueri yang dikelilingi oleh beberapa warna kuning dan satu warna ungu, terlihat ada satu warna kuning dan satu warna ungu sebagai tetangga terdekat. Masuk akal jika kita mengira titik kueri kemungkinan besar berwarna kuning, namun karena *k=1*, k-NN salah memprediksi bahwa titik kueri berwarna sebenarnya adalah warna ungu.

2. Sebaliknya, ketika kita meningkatkan nilai *k*, prediksi kita menjadi lebih stabil karena mayoritas voting/rata-rata, dan dengan demikian, lebih besar kemung-kinannya untuk membuat prediksi yang lebih akurat (hingga titik tertentu).
3. Pada kasus di mana kita mengambil suara mayoritas (misalnya memilih mode dalam masalah klasifikasi) di antara label, kita biasanya menjadikan *k* sebagai angka ganjil untuk mendapatkan tiebreak.

Algoritma k-NN

Asumsikan ada dua kategori, Kategori A dan Kategori B, dan kita menerima titik data baru x1. Di antara kategori berikut, titik data ini termasuk dalam kategori manakah? Algoritma k-NN diperlukan untuk mengatasi masalah seperti ini. Kita cukup menentukan kategori atau kelas suatu dataset dengan bantuan k-NN. Pada Gambar 2, disajikan diagram sebuah data yang terdiri dua kelas A (hijau) dan kelas B (orange). Selain itu ada sebuah data lain (biru) yang terletak diantara dua kelas tersebut. Pertanyaannya masuk kelas apa data tersebut. Setelah diproses dengan k-NN data tersebut dikenali sebagai kelas A. Bagaimana proses dan cara menghitungnya dijelaskan pada bagian selanjutnya
Perhatikan Gambar 10.2 di bawah ini.



Gambar 10.2 Klasifikasi k-NN dengan jarak kelas

Berikut ini algoritma yang dapat menjelaskan bagaimana k-NN bekerja untuk klasifikasi.

1. Tentukan jumlah tetangga (*k*).
   Ini mengubah nilai riil antara 0 dan 1 menjadi nilai lain.
2. Tentukan jarak *Euclidean* antara *k* tetangga.
3. Menggunakan jarak *Euclidean* yang diperoleh, temukan *k* tetangga terdekat.
4. Hitung jumlah titik data di setiap kategori di antara *k* tetangga.
5. Tetapkan titik data baru ke kategori dengan jumlah tetangga terbanyak (mayoritas).
6. Selesai.

Berdasarkan enam langkah tersebut gambar 2 dapat dijelaskan sebagai berikut. Ada tiga warna pada gambar tersebut. Masing-masing adalah warna hijau untuk kategori A dan warna krem untuk kategori B serta warna biru sebagai data baru (non kategori). Misal kita tentukan jumlah k = 3, artinya dicari 3 elemen dari kategori A (hijau) dan kategori B(krem) yang terdekat. Termasuk kategori apakah data baru tersebut? Langkah pertama dihitung jarak masing-masing dari tiap kategori A dan kategori B ke databaru. Selanjutnya

berdasarkan jarak dari tiap-tiap kategori kita cari jarak terdekat. Mayoritas yang terdekat dari kategori A dan kategori B ke data baru, maka data baru termasuk ke dalam kelas mayoritas tersebut. Sebagaimana ditunjukkan pada Gambar 2 data baru masuk ke dalam kelas kategori A.

Menghitung Jarak

Kita telah melihat bahwa algoritma ini malas dan selama waktu pelatihan, yang dilakukannya hanyalah menyimpan semua data yang didapatnya. Semua penghitungan terjadi selama penilaian, yaitu saat kita menerapkan model pada titik data yang tidak terlihat. Kita perlu menentukan k titik data mana dari set pelatihan kita yang paling dekat dengan titik data yang ingin kita prediksi. Data dalam hal ini direpresentasikan dalam sebuah tabel yang terdiri dari n baris dan $m+1$ kolom dengan m kolom pertama adalah atribut yang digunakan untuk memprediksi kolom label dan kolom terakhir juga dikenal sebagai "target". Diasumsikan juga bahwa semua nilai atribut x adalah numerik sedangkan nilai label untuk y bersifat kategorikal, yaitu kita mempunyai masalah klasifikasi.

Berikutnya adalah mendefinisikan fungsi jarak yang menghitung jarak antar titik data. Terutama, ia harus menemukan titik data terdekat dari data pelatihan untuk setiap titik baru.

Jarak Euclidean

Euclidean seringkali merupakan pilihan yang baik untuk fungsi jarak jika datanya numerik. Jika titik data baru kita mempunyai nilai atribut s_1 hingga s_m, kita dapat menghitung jarak $d(s, x_j)$ antara titik s ke titik data mana pun x_j dengan cara

$$d(s, x_j) = \sqrt{(s_1 - x_{j1})^2 + \cdots + (s_m - x_{jm})^2} \quad (1)$$

atau

$$d(s, x) = \sqrt{\sum_{j=1}^{m} (s_j - x_i)^2} \quad (2)$$

Jarak Manhattan

Manhattan ini juga merupakan metrik jarak populer lainnya, yang mengukur nilai absolut antara dua titik dan y_i. Hal ini juga disebut sebagai jarak blok kota karena biasanya divisualisasikan dengan grid, yang menggambarkan bagaimana seseorang dapat bernavigasi dari satu alamat ke alamat lain melalui jalan-jalan kota.

$$d(x, y) = \left(\sum_{i=1}^{m} |x_i - y_i| \right) \quad (3)$$

Jarak Minkowski

Ukuran jarak ini merupakan bentuk umum dari metrik jarak *Euclidean* dan *Manhattan*. Parameter *p* dalam rumus di bawah, memungkinkan pembuatan metrik jarak lainnya. Jarak *Euclidean* diwakili oleh rumus ini ketika *p* sama dengan dua, dan jarak *Manhattan* dilambangkan dengan p sama dengan satu.

$$d(x, y) = \left(\sum_{i=1}^{m} |x_i - y_i| \right)^{1/p} \quad (4)$$

Jarak Hamming

Teknik ini biasanya digunakan dengan vektor

Boolean atau string, mengidentifikasi titik-titik di mana vektor-vektor tersebut tidak cocok. Oleh karena itu, metrik ini juga disebut sebagai metrik yang tumpang tindih. Hal ini dapat direpresentasikan dengan rumus berikut:

$$D_H = \left(\sum_{i=1}^{m} |x_i - y_i| \right) \quad (5)$$

Keterangan
Jika x = y , maka D = 0, untuk x ≠ y, maka D = 1

Keunggulan k-NN

1. Mudah diterapkan
   Mengingat kesederhanaan dan keakuratan algoritma, ini adalah salah satu pengklasifikasi pertama yang akan dipelajari oleh data scientist baru.
2. Beradaptasi dengan mudah
   Saat sampel pelatihan baru ditambahkan, algoritme menyesuaikan dengan memperhitungkan data baru karena semua data pelatihan disimpan ke dalam memori.
3. Sedikit *hyper* parameter
   k-NN hanya memerlukan nilai k dan metrik jarak, yang termasuk rendah jika dibandingkan dengan algoritme pembelajaran mesin lainnya.

Kekurangan k-NN

1. Tidak berskala dengan baik
   Karena k-NN adalah algoritma yang lambat, maka dibutuhkan lebih banyak memori dan penyimpanan data dibandingkan dengan pengklasifikasi lainnya. Hal ini bisa mahal baik dari segi waktu dan uang. Lebih banyak memori dan penyimpanan akan

meningkatkan pengeluaran bisnis dan lebih banyak data memerlukan waktu lebih lama untuk dihitung. Meskipun struktur data yang berbeda, seperti Ball-Tree, telah dibuat untuk mengatasi inefisiensi komputasi, pengklasifikasi yang berbeda mungkin ideal tergantung pada masalah bisnis.

2. Kutukan dimensi
   Algoritma k-NN cenderung menjadi korban kutukan dimensi, yang berarti algoritma tersebut tidak bekerja dengan baik dengan input data berdimensi tinggi. Hal ini terkadang juga disebut sebagai fenomena puncak di mana setelah algoritme mencapai jumlah fitur yang optimal, fitur tambahan meningkatkan jumlah kesalahan klasifikasi, terutama saat sampel ukurannya lebih kecil.
3. Rentan terhadap *overfitting*
   Karena "kutukan dimensi", k-NN juga lebih rentan terhadap overfitting. Meskipun teknik pemilihan fitur dan reduksi dimensi dimanfaatkan untuk mencegah hal ini terjadi, nilai *k* juga dapat memengaruhi perilaku model. Nilai *k* yang lebih rendah dapat membuat data menjadi overfit, sedangkan nilai *k* yang lebih tinggi cenderung "memperhalus" nilai prediksi karena nilai tersebut dirata-ratakan pada area atau lingkungan yang lebih luas. Namun jika nilai k terlalu tinggi maka dapat menyebabkan underfit pada data.

Aplikasi k-NN dalam Machine Learning

Algoritma k-NN telah digunakan dalam berbagai aplikasi, sebagian besar dalam klasifikasi. Beberapa kasus penggunaan ini di antaranya:

1. Pemrosesan awal data (*Data Preprocessing*)

Kumpulan data sering kali memiliki nilai yang hilang, namun algoritme k-NN dapat memperkirakan nilai tersebut dalam proses yang disebut imputasi data yang hilang.

2. Mesin Rekomendasi (*Recommendation Engines*)
   Menggunakan data *clickstream* dari situs web, algoritma k-NN telah digunakan untuk memberikan rekomendasi otomatis kepada pengguna mengenai konten tambahan. Penelitian menunjukkan bahwa pengguna ditugaskan ke grup tertentu, dan berdasarkan perilaku pengguna grup tersebut, mereka diberikan rekomendasi. Namun, mengingat masalah penskalaan pada k-NN, pendekatan ini mungkin tidak optimal untuk kumpulan data yang lebih besar.
3. Keuangan (*Finance*)
   Ini juga telah digunakan dalam berbagai kasus penggunaan keuangan dan ekonomi. Misalnya, satu makalah menunjukkan bagaimana penggunaan k-NN pada data kredit dapat membantu bank menilai risiko pinjaman kepada organisasi atau individu. Ini digunakan untuk menentukan kelayakan kredit pemohon pinjaman. Jurnal lain menyoroti penggunaannya dalam perkiraan pasar saham, nilai tukar mata uang, perdagangan berjangka, dan analisis pencucian uang.
4. Layanan Kesehatan (*Healthcare*)
   k-NN juga telah diterapkan dalam industri kesehatan, membuat prediksi mengenai risiko serangan jantung dan kanker prostat. Algoritma ini bekerja dengan menghitung ekspresi gen yang paling mungkin.
5. Pengenalan Pola (*Pattern Recognition*)
   k-NN juga membantu mengidentifikasi pola, seperti klasifikasi teks dan digit. Hal ini sangat membantu

dalam mengidentifikasi nomor tulisan tangan yang mungkin Anda temukan di formulir atau amplop surat.

Contoh Penerapan k-NN (phyton)

Sebuah produsen mobil baru-baru ini merilis kendaraan baru. Korporasi bertujuan untuk menargetkan orang-orang yang tertarik membeli SUV dengan iklan. Hasilnya, diperoleh kumpulan data untuk topik ini yang berisi informasi dari banyak pengguna melalui jejaring sosial. Dataset memiliki banyak informasi, namun kita akan fokus pada variabel Gaji dan Usia sebagai variabel independen, dan variabel Pembelian sebagai variabel dependen. Data yang didapat seperti ditunjukkan pada Tabel 1. Pada variabel Pembelian tertulis 0 dan 1. Arti dari 0 adalah karyawan tersebut tidak melakukan pembelian, sementara itu angka 1 memiliki arti karyawan melakukan pembelian.

Tabel 10.1 Data gaji karyawan

Nomor_induk	Jenis_Kelamin	Umur	Gaji	Pembelian
23100100	Wanita	20	15000	0
23100101	Wanita	21	20000	0
23100102	Pria	22	16000	0
23100103	Pria	25	50000	0
23100104	Pria	26	45000	0
23100105	Pria	27	45000	0
23100106	Pria	23	35000	0
23100107	Pria	24	35000	1
23100108	Wanita	24	25000	0
23100109	Wanita	25	30000	0
23100110	Wanita	26	32000	0
23100111	Wanita	28	50000	0
23100112	Pria	28	47000	0
23100113	Pria	32	60000	0
23100114	Pria	31	65000	1
23100115	Pria	29	43000	1
23100116	Wanita	41	45000	1
23100117	Pria	43	60000	1
23100118	Pria	43	55000	1
23100119	Wanita	44	50000	1

Langkah-Langkah Penyelesaian

1. Data Pre-processing Step
   Pada tahapan ini yang perlu disiapkan adalah mengimpor library numpy, pandas dan matplotlib.pyplot. Kemudian memanggil dataset yang sudah tersedia didirektori khususnya data dalam format *.csv. Dataset ini pada umum hasil dari pengumpulan di lapangan.

```
#memanggil libraries
import numpy as nm
import matplotlib.pyplot as mtp
import  pandas as pd

#memanggil data
data = pd.read_csv('D:\gaji.csv',sep=";")
data.head(20)
```

   Hasil dari runing code ini ditunjukkan pada Tabel 2

Tabel 10.2 Isi data karyawan

	Nomor_induk	Jenis_Kelamin	Umur	Gaji	Pembelian
0	23100100	Wanita	20	15000	0
1	23100101	Wanita	21	20000	0
2	23100102	Pria	22	16000	0
3	23100103	Pria	25	50000	0
4	23100104	Pria	26	45000	0
5	23100105	Pria	27	45000	0
6	23100106	Pria	23	35000	0
7	23100107	Pria	24	35000	1
8	23100108	Wanita	24	25000	0
9	23100109	Wanita	25	30000	0
10	23100110	Wanita	26	32000	0
11	23100111	Wanita	28	50000	0
12	23100112	Pria	28	47000	0
13	23100113	Pria	32	60000	0
14	23100114	Pria	31	65000	1
15	23100115	Pria	29	43000	1
16	23100116	Wanita	41	45000	1
17	23100117	Pria	43	60000	1
18	23100118	Pria	43	55000	1
19	23100119	Wanita	44	50000	1

Proses selanjutnya setelah memanggil dataset dan telah diketahui isi dari data tersebut maka di lakukan seleksi fitur yang akan digunakan untuk diproses berikutnya. Seleksi fitur dilakukan karena dataset memiliki fitur yang banyak dan tidak semua digunakan. Dalam kasus ini fitur yang digunakan adalah fitur ke 2 (umur), fitur 3 (gaji) dan fitur 4 (pembelian). Perintah yang digunakan untuk seleksi fitur adalah *iloc*[].values. Data dioleh dari dataset **gaji.csv** dan dipilih fitur umur dan gaji sementara itu untuk fitur pembelian adalah sebagai fitur pengelompokan [0,1].

```
#ekstraksi variabel umur, gaji dan pembelian
x = data.iloc[:,[2,3]].values
y = data.iloc[:, 4].values
```

Setelah data siap, proses selanjutnya adalah membagi data menjadi dua bagian, yaitu data latih dan data uji. Perintah yang digunakan *train_test_split()*. Pembagian untuk data latih 75% dan data uji 25% (*test_size = 0.25*). Sehingga dari 20 data yang tersedia di Tabel 1, berubah menjadi data 15 training dan 5 data testing.

```
#membagai data menjadi data latih dan data uji
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size =
0.25, random_state=0)

#penskalaan fitur (gaji dan pembelian)
from sklearn.preprocessing import StandardScaler
st_x = StandardScaler()
x_train = st_x.fit_transform(x_train)
x_test = st_x.transform(x_test)
```

2. Fitting the k-NN algorithm

   Proses untuk mendapatkan pola dilakukan terhadap data latih yang telah dipilih secara acak untuk mendapatkan model klasifikasi k-NN terbaik. Algorima k-NN berdasarkan jumlah data tetangga mayoritas yang terdekat dalam rang *k*. Untuk mengukur kedekatan jarak

ada beberapa formula yang dapat digunakan, yaitu: *Euclidean distance, Manhattan distance, Minkowski distance, Hamming distance*. Dalam contoh ini digunakan jarak yang digunakan adalah *'minkowski'* dengan jumlah ketetanggannya adalah 5 (k=5).

```
from sklearn.neighbors import KNeighborsClassifier
classifier= KNeighborsClassifier(n_neighbors = 5,
metric ='minkowski', p = 2)
classifier.fit(x_train, y_train)
KNeighborsClassifier(algorithm = 'auto', leaf_size = 30,metric =
'minkowski',
metric_params = None, n_jobs = None, n_neighbors = 5, p =
2,weights = 'uniform')
```

3. Predicting the test result

   Pola yang telah didapat saat pelatihan dapat digunakan untuk memprediksi terhadap data baru (*test*), sehingga diketahui termasuk ke dalam kelas apa saja data-data tersebut. Berikut ini perintah yang digunakan dan hasil yang diperoleh.

```
y_pred = classifier.predict(x_test)

array([1, 0, 1, 0, 0], dtype=int64)
```

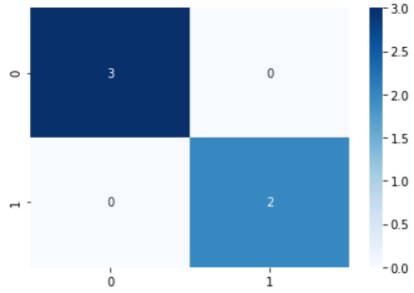
   Dari lima data uji yang diinputkan diperoleh hasil klasifikasi adalah dua data sebagai kelas 1 dan tiga data sebagai kelas 0.

4. Test accuracy of the result

   Akurasi pengujian penting untuk dilakukan agar didapat model klasifikasi yang andal. Untuk memetakan hasil tersebut biasanya digunakan *confusion matrix*.

```
#Creating the Confusion matrix
import seaborn as sns
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred.round())
sns.heatmap(cm,annot=True,cmap="Blues",fmt="d",cbar=True)
```

   Hasil runing code program untuk menampilkan maphing data dengan *confusion matrix* seperti diperlihatkan oleh Gambar 10.3.



Gambar 10.3 Hasil confusion matrix

Hasilnya dapat dijelaskan bahwa tiga data kelas 0 dikenali dengan tepat sebagai kelas 0 dan dua data kelas 1 dikenal dengan tepat sebagai kelas 1. Variabel cm adalah penampung hasil dari *confusion matrix* dan isinya seperti berikut.

```
array([[3, 0],[0, 2]], dtype=int64)
```

Sementara itu untuk menampilkan hasil akurasi klasifikasi bisa juga menggunakan library *accurasy_score*. Perintah berikut ini adalah code untuk menampilkan akurasi klasifikasinya

```
from sklearn.metrics import accuracy_score
ac=accuracy_score(y_test, y_pred.round())
print('accuracy of the model: ',ac)
```

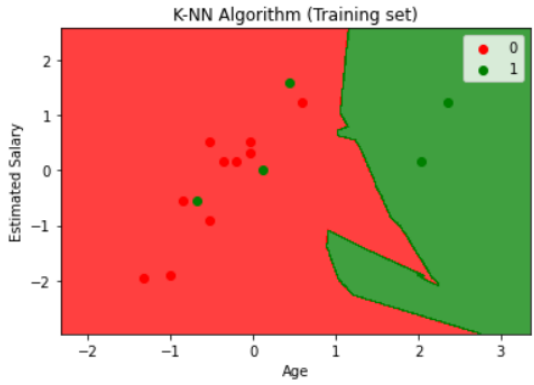
dengan hasil adalah

```
accuracy of the model:  1.0
```

Pelatihan (*training*) adalah proses paling penting dalam mesin pembelajaran. Pembelajaran harus menghasilkan model yang cerdas mengenali data pada tiap kelasnya. Memahami proses klasifikasi untuk tiap data menjadi penting. Oleh karena itu agar mudah dipahami maka perlu ditampilkan secara visual hasil klasifikasi baik saat pelatihan maupun saat pengujian. Pada sub bagian ini disajian *script* pelatihan dan *script* pengujian serta

tampilan gambar hasil pelatihan dan pengujian.

```
#Visualisasi hasil proses pelatihan
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop
= x_set[:, 0].max() + 1, step  = 0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1,
step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(),
x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('red','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
        c = ListedColormap(('red', 'green'))(i), label = j)
mtp.title('K-NN Algorithm (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

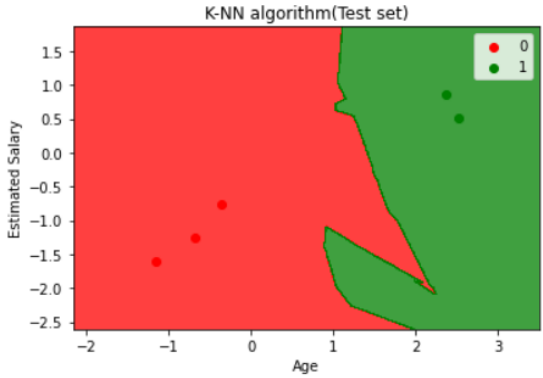


Gambar 10.4 Hasil proses pelatihan

Pada gambar 10.4 tampak ada tiga kelas 1 (hijau) yang terklasifikasi sebagai kelas 0. Jumlah data yang digunakan pada proses pelatihan ini ada 15. Sementara yang salah kelas ada 3, sehingga kesalahannya adalah 3/15 = 20% dan keberhasilannya adalah 80%

```
#Visualisasi hasil pengujian
 from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop
= x_set[:, 0].max() + 1, step  = 0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1,
step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(),
x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('red','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
        c = ListedColormap(('red', 'green'))(i), label = j)
mtp.title('K-NN algorithm(Test set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

Script untuk menampilkan hasil pengujian tersebut bila di runing maka hasil yang diperoleh seperti diperlihatkan pada Gambar 10.5. Dari lima data uji dapat dikelompokan dengan benar (100%), yaitu tiga data sebagai kelas 0 dan dua data sebagai kelas 1.



Gambar 10.5 Hasil proses pengujian