Praktikum 3
Algoritma Klasifikasi KNN

Tujuan

Setelah mengikuti praktikum ini, mahasiswa diharapkan mampu:

1. Mengimplementasikan berbagai algoritma perhitungan jarak.
2. Mengimplementasikan algoritma klasifikasi KNN.
3. Memahami pengaruh parameter *k* pada KNN.

Dasar Teori

Pengukuran jarak pada data digunakan untuk mengetahui perbedaan relatif antara dua data. Beberapa algoritma pembelajaran mesin, seperti K-Nearest Neighbor, Learning Vector Quantization, dan K-Means, memerlukan perhitungan jarak pada salah satu tahapannya. Terdapat beberapa metode perhitungan jarak yang sering digunakan:

1. Manhattan / City Block Distance
   Manhattan distance menghitung jarak dua vektor data berdasarkan panjang total dari proyeksi garis menghubungkan kedua vektor pada masing-masing axis. Rumus dari Manhattan distance adalah :

$$d = \sum_i^n \quad |x_i - y_i|$$

2. Euclidean Distance
   Euclidean distance menghitung jarak dua vektor data berdasarkan panjang dari garis lurus yang menghubungkan kedua vektor. Perhitungan menggunakan akar dari jumlah kuadrat selisih kedua vektor

$$d = \sqrt{\sum_i^n \quad \left(x_i - y_i \quad\right)^2}$$

3. Minkowski Distance
   Minkowski distance merupakan bentuk umum dari fungsi jarak dua vektor

$$d = \left(\sum_{i}^{n} \quad |x_i - y_i|^p \right)^{1/p}$$

Algoritma K-Nearest Neighbor (KNN) merupakan sebuah algoritma klasifikasi yang bertujuan memberikan label kepada data uji berdasarkan label dominan dari *k* data latih yang paling dekat. Secara umum, algoritma KNN bekerja berdasarkan tiga tahapan berikut:

1. Hitung jarak dari data uji ke setiap data latih.
2. Ambil *k* data dengan jarak terkecil.
3. Tentukan kelas data uji menggunakan mayoritas kelas *k* data latih terdekat.

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/6f9306ad21398e
a43cba4f7d537619d0e07d5ae3/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 latih dan data uji menjadi variabel tersendiri. Beri nama variabelnya **label_latih** dan **label_uji.** Fungsi **pop()** pada dataframe berfungsi untuk mengambil suatu kolom dari dataframe, hampir sama dengan fungsi pop() pada struktur data stack.

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

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

```
data_uji.head()
```

3. Menghitung Jarak Euclidean

Tahapan awal dari algoritma KNN adalah perhitungan jarak. Salah satu metode perhitungan jarak yang bisa digunakan adalah jarak Euclidean. Buatlah fungsi bernama **jarakEu** yang berfungsi menghitung jarak euclidean dari dua buah vektor (tanpa kelas)

```
def jarakEu(data1,data2):
  jarak = np.square(data1-data2)
  jarak = np.sum(jarak)
  return np.sqrt(jarak)
```

Perhitungan jarak Euclidean sangat sederhana karena memanfaatkan operasi vektor menggunakan *numpy* sehingga tidak perlu melakukan *looping* secara eksplisit. Ingat, input dari **jarakEu** merupakan dataframe yang mewarisi sifat numpy array. Operasi pertama pada perhitungan jarak Euclidean adalah mengurangi data1 dengan data2. Operasi pengurangan pada numpy dilakukan secara *element-wise* atau per elemen. Setelah itu, hasil pengurangannya dikuadratkan menggunakan fungsi **square** pada numpy. Selanjutnya, semua elemen hasil kuadrat tadi dijumlahkan menggunakan fungsi **sum** pada numpy. Terakhir, hasil penjumlahkan diakar kuadrat menggunakan fungsi **sqrt** pada numpy.

Uji fungsi **jarakEu** untuk menghitung jarak antara data latih pertama dengan data uji pertama.

```
jarak = jarakEu(data_latih.iloc[0],data_uji.iloc[0])
print(jarak)
```

Fungsi **iloc[x]** pada sebuah dataframe digunakan untuk mendapatkan nilai dataframe pada baris dengan index x.

4. Algoritma KNN

Setelah fungsi perhitungan jarak selesai dibuat, tahapan selanjutnya adalah mengimplementasikan algoritma KNN. Implementasikan algoritma KNN sesuai dengan tahapan-tahapan yang dijelaskan pada bagian teori. Buatlah fungsi dengan nama **knn** yang menerima input berupa *k*, sekumpulan data latih dan labelnya, serta sebuah data uji.

```
from collections import Counter
def knn(k,datalatih,labellatih,datauji):
  jarak = np.array([jarakEu(datalatih.iloc[x],datauji) for x in
range(datalatih.shape[0])])
  indeks_k_minimum = jarak.argsort()[:k]
  k_kelas = labellatih.iloc[indeks_k_minimum].to_numpy()
  counter = Counter(k_kelas)
  kelas_uji = counter.most_common(1)[0][0]
  return kelas_uji
```

Algoritma KNN diawali dengan penghitungan jarak antara satu data uji dengan semua data latih. Pada fungsi tersebut, perhitungan jarak memanfaatkan list comprehension. Nilai indeks dari jarak terkecil didapatkan melalui fungsi argsort pada numpy mendapatkan indeks dari jarak yang telah diurutkan. Penambahan indeks `[:k]` menandakan diambil dari *k* data dengan jarak terkecil. Indeks tersebut digunakan untuk mendapatkan kelas dari *k* data dengan jarak terkecil. Proses majority voting memanfaatkan Counter yang merupakan dictionary yang berisi frekuensi obyek yang disimpan.

Lakukan pengujian fungsi **knn** untuk menentukan kelas dari data uji pertama

```
hasil = knn(3,data_latih,label_latih,data_uji.iloc[0])
print(hasil)
```

5. KNN pada Scikit-learn

Library scikit-learn memiliki algoritma KNN yang siap untuk digunakan. Sebagai pengenalan, Anda akan melakukan klasifikasi data yang sama menggunakan KNN yang tersedia pada library scikit-learn

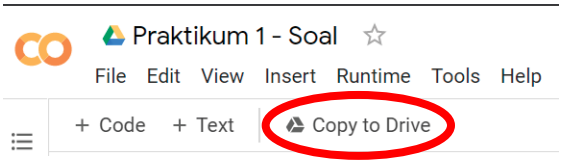
```
from sklearn.neighbors import KNeighborsClassifier
KNN = KNeighborsClassifier(n_neighbors=3)
KNN.fit(data_latih,label_latih)
kelas = KNN.predict([data_uji.iloc[0]])
print(kelas)
```

Tugas

1. Buatlah fungsi bernama **jarakCity** yang berfungsi menghitung jarak City Block antara dua vektor.
2. Buatlah sebuah fungsi bernama **knn_multi** yang dapat menentukan kelas dari **semua** data uji (Fungsi **knn** yang sudah Anda buat hanya dapat menentukan kelas dari sebuah data uji. Anda dapat memanfaatkan fungsi **knn** yang telah ada
3. Buatlah fungsi bernama **cek_hasil** yang berfungsi membandingkan hasil klasifikasi dengan label sebenarnya. Fungsi ini menerima input 2 buah list, yaitu list label hasil klasifikasi dan list label data uji yang sebenarnya. Output dari fungsi ini adalah integer, yang menunjukkan berapa banyak label hasil klasifikasi yang sama dengan label data uji sebenarnya.
4. Jalankan fungsi **knn_multi** dengan nilai k=3, k=7,k=9 dan k=13. Amati hasil dari fungsi **cek_hasil** dengan nilai k tersebut. Apa kesimpulan Anda?

Petunjuk pengerjaan soal:

1. Klik link berikut, pastikan Anda login menggunakan **akun student UB**. https://colab.research.google.com/drive/1lFs3BNNsEaaDsoogO2FoD-bo0TEggpxa?usp=sharing
2. Klik tombol Copy to Drive



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