13522103 - Steven Tjhia

K-Nearest-Neighbors

1. Jelaskan cara kerja dari algoritma tersebut! (boleh dalam bentuk *pseudocode* ataupun narasi)
   ⇒ Pertama, model menyimpan dataset latihan (*X_train* dan *y_train*). Kemudian ketika melakukan prediksi, “titik” masukan diukur jaraknya terhadap sebanyak *k* “titik” terdekat dari data latihan. Setelah itu diambil kesimpulan dengan melihat hasil terbanyak dari klasifikasi “titik” data latihan tersebut.
   Pada model KNN yang dibuat sendiri, saya menambahkan implementasi *tie-breaker*, jika jumlah kesimpulan klasifikasi sama, maka akan dipilih klasifikasi dengan jarak rata-rata paling kecil.

2. Bandingkan hasil evaluasi model from scratch dan *library*, bagaimana hasil perbandingannya? Jika ada perbedaan, jelaskan alasannya!

Akurasi [holdout 80-20]	
Scratch	***Library***
0.702	0.725
Akurasi [k-fold 5]	
Scratch	***Library***
0.727	0.719
Waktu Eksekusi	
Scratch	***Library***
9.6 detik	0.05 detik

Berdasarkan data tersebut, akurasi [holdout 80-20] yang dihasilkan dengan menggunakan *library* lebih besar 0.025, tetapi akurasi [k-fold 5] yang dihasilkan dari model KNN yang dibuat dari *scratch* lebih besar 0.08. Perbedaan akurasi tersebut tidak terlalu signifikan, tetapi model KNN yang dibuat dari *scratch* memerlukan waktu eksekusi yang jauh lebih lama yaitu 9.6 detik dibandingkan dengan *library* yaitu 0.05

detik. Hal tersebut dikarenakan model KNN dengan *library sklearn* memiliki optimasi kecepatan dan penggunaan memori, sedangkan implementasi *scratch* yang dibuat belum mempertimbangkan hal tersebut.

3. Jelaskan *improvement* apa saja yang bisa Anda lakukan untuk mencapai hasil yang lebih baik dibandingkan dengan hasil yang Anda punya saat ini! *Improvement* yang dimaksud tidak terbatas pada bagaimana algoritma diimplementasikan, namun juga mencakup tahap sebelum *modeling and validation*.

   ⇒ Data preprocessing: karena jarak antar-"titik" sangat berpengaruh pada model KNN, maka akan sangat disarankan untuk melakukan *scaling* pada data terlebih dahulu dengan menggunakan beberapa *tools* seperti StandardScaler ataupun RobustScaler.

   ⇒ Hyperparameter tuning: memilih nilai *k* yang sesuai dapat berdampak besar pada prediksi yang akan dihasilkan. Pada saat percobaan nilai *k* yang digunakan adalah 2, jika menggunakan nilai *k* lebih dari 2, maka kemungkinan besar akurasi yang dihasilkan model dapat meningkat, namun tentunya nilai *k* tidak boleh terlalu besar juga, jadi benar-benar harus dipilih nilai *k* yang sangat tepat agar menghasilkan hasil yang maksimal.

   ⇒ Algoritma: menerapkan algoritma perhitungan jarak yang lebih efektif dan efisien dengan melakukan optimasi pada kecepatan dan penggunaan memori akan membuat model menjadi lebih efektif dan efisien.