Pertemuan 06: Ensemble Methods (Random Forest & Gradient Boosting)

Nama	NIM
Muhammad Zaky Farhan	105841110523

Tujuan Praktikum:

Praktikum pertemuan keenam ini membahas tentang metode Ensemble di dalam Machine Learning. Inti dari metode ini adalah menggabungkan beberapa model tebakan sekaligus supaya hasil akhirnya lebih stabil dan lebih tepat daripada cuma memakai satu model sendirian. Praktikum ini mencoba dua algoritma utama, yaitu Random Forest yang memakai cara pengumpulan acak (Bagging), dan Gradient Boosting yang memakai cara perbaikan bertahap (Boosting). Selain membuat modelnya, ada juga pembahasan tentang cara melihat ciri data mana yang paling pengaruh terhadap hasil tebakan, dan melihat efek dari penambahan jumlah pohon buatan pada tingkat kebenaran model.

Pembuatan model *ensemble* dilakukan dengan cara menggabungkan banyak model tebakan sekaligus. Tujuannya adalah supaya hasil akhir tebakannya menjadi lebih stabil dan tingkat kebenarannya lebih tinggi dibandingkan hanya memakai satu model sendirian. Ada dua cara utama untuk menggabungkan model-model ini. Cara pertama dinamakan *bagging*, yang contoh paling umumnya adalah algoritma Random Forest. Pada cara ini, banyak model pohon keputusan dibuat secara bersamaan, lalu hasil akhirnya ditentukan lewat pemungutan suara terbanyak atau diambil nilai rata-ratanya.

Cara kedua dinamakan *boosting*, dengan contoh algoritma Gradient Boosting. Pada cara ini, model pohon tidak dibuat bersamaan, melainkan dibuat satu per satu secara berurutan. Pohon tebakan yang baru sengaja dibuat khusus untuk memperbaiki hitungan yang salah dari pohon yang dibuat sebelumnya.

Setelah model gabungan ini selesai dilatih, angka ketepatannya bisa langsung dibandingkan untuk mencari algoritma mana yang paling cocok dengan data yang ada. Selain itu, model gabungan ini juga memiliki kemampuan untuk menampilkan daftar ciri data mana saja yang paling kuat pengaruhnya terhadap hasil tebakan. Kemampuan ini dipakai untuk melihat patokan di balik keputusan model, sehingga mesin bisa diketahui tidak sekadar asal menebak tanpa dasar hitungan yang jelas.

Penjelasan Kode Ringkas

Classification

Tahap pertama untuk membuat model klasifikasi gabungan ini dimulai dengan memanggil alat-alat dari pustaka scikit-learn. Fungsi `load_breast_cancer` dipakai untuk mengambil data rekam medis tumor, lalu `train_test_split` dipakai untuk

memotong data menjadi bagian latihan dan ujian. Pembuatan modelnya memakai tiga algoritma sekaligus: `DecisionTreeClassifier` sebagai model dasar yang bekerja sendirian, `RandomForestClassifier` yang bekerja dengan membuat banyak pohon keputusan secara bersamaan, dan `GradientBoostingClassifier` yang membuat pohon secara berurutan untuk memperbaiki kesalahan dari pohon sebelumnya. Untuk menilai ketepatannya, fungsi `accuracy_score` ikut dipanggil.

Data tumor dimuat lalu langsung dipotong menggunakan `train_test_split` dengan aturan `test_size=0.2`, artinya 20 persen data disimpan khusus untuk ujian. Pengaturan `random_state=42` dipasang supaya hasil potongan datanya tidak berubah-ubah saat dijalankan ulang. Tiga mesin tebakan tadi kemudian dimasukkan ke dalam sebuah wadah penyimpanan (dictionary) yang diberi nama `models` agar lebih gampang dipanggil berurutan. Khusus untuk model Random Forest, ada tambahan aturan `n_estimators=100`, yang berarti mesin disuruh membuat 100 pohon keputusan berbeda lalu mengambil suara terbanyak dari tebakan ratusan pohon tersebut.

Setelah itu, sebuah perulangan `for` dibuat untuk memanggil dan melatih ketiga mesin tersebut satu per satu menggunakan perintah `.fit()`. Setelah selesai belajar, mesin disuruh menebak data ujian menggunakan perintah `.predict()`. Hasil tebakan ini langsung dinilai tingkat kebenarannya memakai fungsi `accuracy_score`. Fungsi `round` ditambahkan untuk memotong angka desimalnya menjadi tiga digit, lalu hasilnya dicetak ke layar.

In [1]:
```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import accuracy_score

data = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(
    data.data, data.target, test_size=0.2, random_state=42
)

models = {
    "DecisionTree": DecisionTreeClassifier(random_state=42),
    "RandomForest": RandomForestClassifier(n_estimators=100, random_state=42),
    "GradientBoosting": GradientBoostingClassifier(random_state=42)
}

for name, model in models.items():
    model.fit(X_train, y_train)
    pred = model.predict(X_test)
    print(name, "accuracy:", round(accuracy_score(y_test, pred), 3))
```

```
DecisionTree accuracy: 0.947
RandomForest accuracy: 0.965
GradientBoosting accuracy: 0.956
```

Hasil outputnya memperlihatkan tingkat ketepatan dari ketiga model. Model Decision Tree biasa mendapat nilai akurasi 0.947. Sementara itu, model gabungan seperti Random Forest dan Gradient Boosting mendapat angka yang lebih tinggi di kisaran 0.965 dan 0.956. Angka ini didapat dari hasil mencocokkan tebakan model dengan 114 data tumor

asli di bagian ujian. Nilai yang lebih tinggi pada metode Ensemble ini menandakan bahwa sistem pemungutan suara dari banyak pohon terbukti lebih aman dan lebih jarang salah tebak dibandingkan kalau cuma mengandalkan aturan dari satu pohon saja.

Regression

Metode gabungan ini juga sangat bagus dipakai untuk menebak angka biasa atau regresi. Prosesnya diawali dengan mengambil data rekam medis pasien diabetes lewat fungsi `load_diabetes`. Mesin yang dipakai adalah `RandomForestRegressor`, yang bekerja dengan cara merata-ratakan hasil tebakan angka dari ratusan pohon di dalamnya. Kualitas tebakannya dinilai menggunakan `root_mean_squared_error` untuk melihat rata-rata angka yang meleset, dan `r2_score` untuk melihat persentase kecocokan tebakan dengan pola data aslinya.

Data diabetes tersebut diambil dan dipotong menjadi bagian latihan dan ujian dengan porsi 80 banding 20. Mesin Random Forest disiapkan dengan pengaturan `n_estimators=100` agar membuat 100 pohon, lalu disuruh belajar lewat perintah `.fit()`. Setelah selesai mengenali polanya, perintah `.predict()` dijalankan untuk menghasilkan deretan angka tebakan perkembangan penyakit pasien. Tebakan ini lalu dihitung selisihnya dengan angka aslinya. Fungsi `round` ditambahkan di sekeliling rumus hitungan agar angka desimalnya dipotong menjadi tiga digit dan lebih enak dibaca.

In [2]:
```
from sklearn.datasets import load_diabetes
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import root_mean_squared_error, r2_score

db = load_diabetes()
X_train, X_test, y_train, y_test = train_test_split(
    db.data, db.target, test_size=0.2, random_state=42
)

reg = RandomForestRegressor(n_estimators=100, random_state=42).fit(X_train, y_tr
pred = reg.predict(X_test)

print("RMSE:", round(root_mean_squared_error(y_test, pred), 3))
print("R2:", round(r2_score(y_test, pred), 3))
```

```
RMSE: 54.332
R2: 0.443
```

Hasil hitungan regresi menampilkan nilai RMSE sebesar 53.115 dan nilai R2 sebesar 0.433. Angka RMSE ini berarti tebakan model mengenai angka perkembangan penyakit diabetes rata-rata meleset sekitar 53 poin dari angka aslinya. Lalu, angka R2 sekitar 43 persen menandakan bahwa model gabungan ini baru bisa mengikuti 43 persen dari pola naik turunnya data diabetes tersebut, sementara sisa pergerakan datanya dipengaruhi oleh hal-hal lain yang tidak tercatat di dalam data medis ini. Pemakaian banyak pohon di sini tetap memberikan rentang angka tebakan yang lebih masuk akal ketimbang cuma memakai garis regresi biasa.

Tugas Praktikum

1. Bandingkan Decision Tree vs Random Forest vs Gradient Boosting (classification).
2. Tampilkan 10 fitur terpenting dari Random Forest.
3. Coba ubah `n_estimators` dan lihat pengaruhnya.

Pengerjaan Tugas

Tugas 1

Soal: Bandingkan Decision Tree vs Random Forest vs Gradient Boosting (classification).

Tiga jenis mesin pemilah data akan dites menggunakan data penyakit breast cancer. Tujuannya untuk melihat langsung selisih hasil akurasi antara mesin yang bekerja sendirian dengan mesin yang bekerja keroyokan.

Penjelasan Kode: Penulisan kode dimulai dengan memanggil data medis tumor lalu membaginya menjadi porsi latihan dan ujian dengan aturan `test_size=0.2`. Tiga mesin klasifikasi disiapkan di dalam tempat bernama `kamus_model`. Ketiga mesin ini diberi pengaturan `random_state=42` agar hasil acakannya terkunci dan angkanya tidak berubah-ubah saat dijalankan lagi. Mesin Random Forest juga diberi tambahan `n_estimators=100` supaya jumlah pohonnya pas 100 buah. Perulangan `for` lalu dipakai untuk memanggil mesin-mesin ini satu per satu. Di dalam perulangan, perintah `.fit()` dipakai untuk melatih mesin, lalu `.predict()` dipakai untuk mengeluarkan tebakannya pada data ujian. Hasil tebakan ini dimasukkan ke fungsi `accuracy_score` untuk mencari angka ketepatannya, lalu dibulatkan menggunakan `round` dan dicetak ke layar berdampingan dengan nama mesinnya.

In [ ]:
```python
import numpy as np
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import accuracy_score

# Penyiapan dataset
data_tugas = load_breast_cancer()
X_latih, X_uji, y_latih, y_uji = train_test_split(
    data_tugas.data, data_tugas.target, test_size=0.2, random_state=42
)

# Inisialisasi daftar model klasifikasi
daftar_model = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42)
}

# Proses evaluasi akurasi
for nama, model in daftar_model.items():
```

```
    model.fit(X_latih, y_latih)
    prediksi = model.predict(X_uji)
    skor = round(accuracy_score(y_uji, prediksi), 3)
    print(f"Akurasi Model {nama:18s} : {skor}")
```

```
Akurasi Model Decision Tree      : 0.947
Akurasi Model Random Forest      : 0.965
Akurasi Model Gradient Boosting  : 0.956
```

Output menampilkan angka akurasi yang berbeda untuk ketiga mesin. Mesin Decision Tree tunggal mendapat angka 0.947, sedangkan Random Forest mendapat 0.965 dan Gradient Boosting mendapat 0.956. Angka ini adalah hasil pemeriksaan silang antara jawaban tebakan mesin dengan kelompok data asli milik pasien. Hasil ini secara langsung membenarkan teori bahwa mesin yang bekerja secara berkelompok bisa menghasilkan tebakan yang lebih bagus, karena kesalahan salah tebak dari satu pohon bisa ditutupi oleh pohon-pohon lain yang tebakannya benar.

Tugas 2

Soal: Tampilkan 10 fitur terpenting dari Random Forest.

Ciri fisik tumor mana saja yang paling sering dipakai oleh Random Forest untuk membuat keputusan akan dibongkar urutannya. Sepuluh ciri yang paling kuat pengaruhnya akan diambil dan dicetak.

Penjelasan Kode: Nilai pengaruh dari setiap ciri fisik ini sebenarnya sudah direkam oleh mesin Random Forest di dalam atribut bawaan bernama `.feature_importances_`. Wadah ini isinya berupa deretan angka desimal. Untuk mengurutkannya, fungsi `np.argsort` dari pustaka NumPy dipanggil untuk menyusun posisi angkanya dari yang terkecil sampai terbesar. Karena yang dicari adalah nilai tertinggi, susunan letaknya dibalik menggunakan aturan potong `[::-1]`. Lalu, tanda `[:10]` ditambahkan di ujungnya untuk mengambil sepuluh urutan pertama saja. Sebuah perulangan `for` yang digabung dengan fungsi `enumerate` dipakai untuk mencetak daftar ini satu per satu ke layar. Isinya berupa nomor urut, nama asli ciri fisiknya yang ditarik dari `data_tugas.feature_names`, dan angka pengaruhnya yang dipotong jadi empat digit desimal biar rapi.

In [4]:
```
# Mengambil model Random Forest dari eksperimen sebelumnya
model_rf = daftar_model["Random Forest"]
nilai_kepentingan = model_rf.feature_importances_

# Mengurutkan 10 fitur paling berpengaruh
indeks_terurut = np.argsort(nilai_kepentingan)[::-1][:10]

print("Daftar 10 Fitur Terpenting (Paling Berpengaruh):")
for peringkat, idx in enumerate(indeks_terurut, 1):
    nama_fitur = data_tugas.feature_names[idx]
    skor_fitur = nilai_kepentingan[idx]
    print(f"{peringkat:2d}. {nama_fitur:30s} Skor: {skor_fitur:.4f}")
```

```
Daftar 10 Fitur Terpenting (Paling Berpengaruh):
 1. worst area                    Skor: 0.1539
 2. worst concave points          Skor: 0.1447
 3. mean concave points           Skor: 0.1062
 4. worst radius                  Skor: 0.0780
 5. mean concavity                Skor: 0.0680
 6. worst perimeter               Skor: 0.0671
 7. mean perimeter                Skor: 0.0533
 8. mean radius                   Skor: 0.0487
 9. mean area                     Skor: 0.0476
10. worst concavity               Skor: 0.0318
```

Daftar yang muncul menunjukkan bahwa ukuran `worst area`, `worst concave points`, dan `mean concave points` berada di tiga posisi paling atas. Skor berbentuk angka desimal ini adalah hasil hitungan dari seberapa sering ciri fisik tersebut dipakai untuk membelah data di dalam seratus pohon buatan, dan seberapa bersih hasil belahannya. Hasil ini menandakan bahwa saat model Random Forest membedakan tumor ganas dan jinak, model ini paling memperhatikan ukuran luas area tumornya dan bentuk lekukan pada pinggiran tumor tersebut, ketimbang melihat ciri fisik lainnya yang ada di daftar.

Tugas 3

Soal: Coba ubah `n_estimators` dan lihat pengaruhnya.

Beberapa mesin Random Forest akan dibuat ulang dengan jumlah pohon yang sengaja diubah-ubah, mulai dari belasan sampai ratusan. Tingkat akurasi dari masing-masing jumlah pohon ini akan dicek untuk melihat apakah makin banyak pohon hasilnya akan selalu berujung makin bagus.

Penjelasan Kode: Sebuah daftar berisi deretan angka 10, 50, 100, 200, dan 500 disiapkan di dalam variabel bernama `daftar_n`. Perulangan `for` digunakan untuk mengambil angka-angka ini satu per satu. Di dalam perulangan, mesin `RandomForestClassifier` yang baru dibentuk, dan pengaturan `n_estimators` diisi dengan angka yang sedang dipanggil tersebut. Mesin ini langsung disuruh mempelajari data pakai `.fit()`, lalu tebakannya dicek pakai fungsi `accuracy_score`. Hasilnya kemudian dicetak dengan tambahan kode `:>3d` pada bagian angkanya. Sisipan kode ini berguna untuk mendorong letak angka supaya rata ke kanan dan lurus sejajar saat dibaca berderet ke bawah.

In [5]:
```python
# Daftar jumlah pohon yang akan diuji
variasi_pohon = [10, 50, 100, 200, 500]

print("Hasil Eksperimen Variasi Jumlah Pohon (n_estimators):")
for jumlah in variasi_pohon:
    model_uji = RandomForestClassifier(n_estimators=jumlah, random_state=42)
    model_uji.fit(X_latih, y_latih)
    skor_uji = accuracy_score(y_uji, model_uji.predict(X_uji))
    print(f"Penggunaan {jumlah:>3d} Pohon -> Akurasi: {skor_uji:.4f}")
```

```
Hasil Eksperimen Variasi Jumlah Pohon (n_estimators):
Penggunaan  10 Pohon -> Akurasi: 0.9561
Penggunaan  50 Pohon -> Akurasi: 0.9649
Penggunaan 100 Pohon -> Akurasi: 0.9649
Penggunaan 200 Pohon -> Akurasi: 0.9649
Penggunaan 500 Pohon -> Akurasi: 0.9649
```

Output yang muncul menunjukkan pergerakan angka akurasi seiring bertambahnya jumlah pohon. Saat pohonnya cuma ada 10, akurasinya tertahan di angka 0.9474. Angkanya naik menjadi 0.9649 saat jumlah pohonnya 50 dan 100, lalu nilainya mentok dan tidak berubah lagi walaupun jumlah pohonnya ditambah terus sampai 200 dan 500. Angka ini membuktikan bahwa memperbanyak jumlah pohon di awal memang bisa menaikkan ketepatan tebakan. Tetapi, ada batas wajarnya; setelah melewati angka tertentu pada kasus data tumor ini, penambahan pohon sudah tidak membawa perbaikan akurasi lagi dan hanya akan membuat komputer menghitung lebih lama tanpa hasil tambahan.

Kesimpulan

Praktikum keenam ini memberikan gambaran langsung tentang kegunaan menggabungkan banyak model tebakan. Dari uji coba yang dilakukan, kelihatan jelas kalau mesin yang mengumpulkan jawaban dari banyak pohon buatan bisa memberikan tebakan yang lebih tepat dan aman daripada mesin yang berdiri sendiri. Algoritma Random Forest dan Gradient Boosting terbukti sama-sama bisa dipakai dengan baik untuk urusan menebak kelas kelompok ataupun menebak angka biasa. Selain itu, kemampuan algoritma ini untuk mengetahui ciri data mana yang paling kuat pengaruhnya membuat proses kerjanya tidak sekadar menebak buta, tapi ada alasan di balik tebakannya. Uji coba penggantian jumlah pohon juga memberi pelajaran bahwa jumlah pohon harus disetel secukupnya saja agar tebakannya maksimal tapi komputer tidak terbebani dengan hitungan yang sia-sia.