# Cascade Classify


| | |
|---|---|
| Narendra Arkan Putra Darmawan | 1313621043 |
| Muhammad Ramadhan Putra Pratama | 1313621038 |

# 1. Fungsi calculate_sumless_mean(data)

```
function calculate_sumless_mean(data)
    averages = Matrix{Float16}(undef, 3, 4)

    for (i, val) in enumerate(1:3)
        filtered_data = data[data[:, 5] .== val, :]

        for col in 1:4
            n = 0
            mean = 0.0

            for x in filtered_data[:, col]
                n += 1
                delta = x - mean
                mean += delta / n
            end

            averages[i, col] = mean
        end
    end
    return averages
end
```

- Fungsi ini menghitung rata-rata setiap fitur (kolom) untuk setiap kelas (nilai 1 hingga 3).
- Variabel averages adalah matriks yang menyimpan hasil perhitungan rata-rata.
- Data difilter berdasarkan kelas, dan rata-rata dihitung dengan metode Welford untuk menghindari sensitivitas terhadap urutan data.

# 2. Fungsi predict_class(data, averages)

```
function predict_class(data, averages)
    predicted_classes = Matrix{Float16}(undef, size(data, 1), 4)

    for row in axes(data, 1)

        for col in axes(data, 2)
            distances = zeros(Float16, 3)

            for i in 1:3
                distances[i] = abs(data[row, col] - averages[i, col])
            end

            predicted_classes[row, col] = argmin(distances)
        end
    end

    return predicted_classes
end
```

- Fungsi ini memprediksi kelas untuk setiap data dalam dataset.
- Menggunakan algoritma Euclidean dengan membandingkan jarak antara nilai fitur dan rata-rata yang telah dihitung.
- Hasil prediksi disimpan dalam matriks predicted_classes.

# 3. Fungsi compare_classes(predicted_classes, actual_classes)

```
function compare_classes(predicted_classes, actual_classes)
    comparison = Matrix{Bool}(undef, size(predicted_classes, 1), 4)

    for row in axes(predicted_classes, 1)

        for col in axes(predicted_classes, 2)
            comparison[row, col] = predicted_classes[row, col] == actual_classes[row]
        end
    end

    return comparison
end
```

- Fungsi ini membandingkan hasil prediksi kelas dengan kelas sebenarnya.
- Membuat matriks comparison yang berisi nilai boolean apakah prediksi benar atau salah.

# 4. Fungsi calculate_accuracy(comparison)

```
function calculate_accuracy(comparison)
    accuracy = zeros(Float32, 4)

    for i in 1:4
        accuracy[i] = mean(comparison[:, i])
    end

    return accuracy
end

```

- Fungsi ini menghitung akurasi untuk setiap fitur berdasarkan matriks perbandingan.
- Menghasilkan nilai akurasi untuk masing-masing fitur.

# 5. Fungsi filter_correct_prediction(data, comparison)

```
function filter_correct_prediction(data, comparison, accuracies)
    correct_size = 0
    incorrect_size = 0
    for row in axes(data, 1)
        if comparison[row, argmax(accuracies)] == true
            correct_size += 1
        else
            incorrect_size += 1
        end
    end

    correct_data = Matrix{Float16}(undef, correct_size, 5)
    incorrect_data = Matrix{Float16}(undef, incorrect_size, 5)
    incorrect_comparison = zeros(Bool, incorrect_size, 4)
    i = 1
    j = 1
    for row in axes(data, 1)
        if comparison[row, argmax(accuracies)] == true
            for col in 1:5
                correct_data[i, col] = data[row, col]
            end
            i += 1
        else
            for col in 1:5
                incorrect_data[j, col] = data[row, col]
                incorrect_comparison[j, :] = comparison[row, :]
            end
            j += 1
        end
    end

    return correct_data, incorrect_data, incorrect_comparison
end
```

- Fungsi ini memisahkan data yang diprediksi dengan benar dan salah.
- Mengembalikan dua matriks terpisah: correct_data dan incorrect_data

# 6. Fungsi correct_prediction_sumless_mean(data, col)

```
function correct_prediction_sumless_mean(data, col)
    averages = Matrix{Float16}(undef, 3, 1)

    for (i, val) in enumerate(1:3)
        filtered_data = data[data[:, 5] .== val, :]
        n = 0
        mean = 0.0

        for x in filtered_data[:, col]
            n += 1
            delta = x - mean
            mean += delta / n
        end

        averages[i, 1] = mean
    end

    return averages
end

```

- Fungsi ini menghitung rata-rata untuk suatu fitur dari data yang diprediksi dengan benar.
- Digunakan untuk menghasilkan rata-rata baru berdasarkan data yang diprediksi dengan benar.

# 7. Fungsi calculate_correct_prediction_averages(data, comparison, accuracies)

```
function calculate_correct_prediction_averages(data, comparison, accuracies)
    averages = Matrix{Float16}(undef, 3, 4)
    temp_accuracies = copy(accuracies)

    x1, remainder_data, remainder_comparison = filter_correct_prediction(data, comparison, temp_accuracies)
    x1_averages = correct_prediction_sumless_mean(x1, argmax(temp_accuracies))
    averages[:, argmax(temp_accuracies)] = x1_averages
    temp_accuracies[argmax(temp_accuracies)] = 0.0

    x2, remainder_data, remainder_comparison = filter_correct_prediction(remainder_data, remainder_comparison, temp_accuracies)
    x2_averages = correct_prediction_sumless_mean(x2, argmax(temp_accuracies))
    averages[:, argmax(temp_accuracies)] = x2_averages
    temp_accuracies[argmax(temp_accuracies)] = 0.0

    x3, remainder_data, remainder_comparison = filter_correct_prediction(remainder_data, remainder_comparison, temp_accuracies)
    x3_averages = correct_prediction_sumless_mean(x3, argmax(temp_accuracies))
    averages[:, argmax(temp_accuracies)] = x3_averages
    temp_accuracies[argmax(temp_accuracies)] = 0.0

    x4, remainder_data, remainder_comparison = filter_correct_prediction(remainder_data, remainder_comparison, temp_accuracies)
    x4_averages = correct_prediction_sumless_mean(x4, argmax(temp_accuracies))
    averages[:, argmax(temp_accuracies)] = x4_averages
    temp_accuracies[argmax(temp_accuracies)] = 0.0

    return averages
end
```

- Fungsi ini menghasilkan rata-rata baru berdasarkan data yang diprediksi dengan benar.
- Menggunakan 2 fungsi sebelumnya untuk memfilter data dan menghitung rata-rata berdasarkan fitur.
- Digunakan untuk pembaruan averages setelah iterasi pertama klasifikasi.

# 8. Pemanggilan Fungsi dan Eksekusi

```
raw_data = deserialize(open("data_9m.mat", "r"))
println("Raw Data: ")
display(raw_data)

averages = calculate_sumless_mean(raw_data)
println("\nAverages: ")
display(averages)

predicted_classes = predict_class(raw_data[:, 1:4], averages)
println("\nPredicted Classes:")
display(predicted_classes)

comparison = compare_classes(predicted_classes, raw_data[:, 5])
println("\nCorrect Predictions: ")
display(comparison)

accuracies = calculate_accuracy(comparison)
println("\nFeature 1 Accuracy: $(accuracies[1] * 100)%")
println("Feature 2 Accuracy: $(accuracies[2] * 100)%")
println("Feature 3 Accuracy: $(accuracies[3] * 100)%")
println("Feature 4 Accuracy: $(accuracies[4] * 100)%")

correct_prediction_averages = calculate_correct_prediction_averages(raw_data, comparison, accuracies)
println("\nNew Averages: ")
display(correct_prediction_averages)
```

- Data yang dideserialize dari file "data_9m.mat" dibaca dan digunakan dalam analisis.
- Setelah melakukan klasifikasi, averages baru dihitung berdasarkan data yang diprediksi dengan benar

# 9. Output Program

```
Raw Data:
9830400×5 Matrix{Float16}:
  5.1      3.5      1.4      0.2    1.0
  4.9      3.0      1.4      0.2    1.0
  4.7      3.2      1.3      0.2    1.0
  4.6      3.1      1.5      0.2    1.0
  ⋮
 25.25    -1.852   -7.69   -11.695  3.0
  3.094    1.225   -6.336   13.93   3.0
 12.44   -10.96   -14.85   -13.12   3.0
 -1.461  -10.664   24.47   -10.44   3.0
```

- Hasil deserialize data mentah

```
Averages:
3×4 Matrix{Float16}:
 3.229  1.777  0.541  -2.166
 4.492  1.169  3.797  -0.517
 5.754  0.92   3.963  -0.3533
```

- Rata-rata atau μ (mu) dari tiap features dan class

```
Predicted Classes:
9830400×4 Matrix{Float16}:
 2.0  1.0  1.0  3.0
 2.0  1.0  1.0  3.0
 2.0  1.0  1.0  3.0
 2.0  1.0  1.0  3.0
 ⋮
 3.0  3.0  1.0  1.0
 1.0  2.0  1.0  3.0
 3.0  3.0  1.0  1.0
 1.0  3.0  3.0  1.0
```

- Hasil prediksi class berdasarkan euclidean distance dari tiap features

# 10. Output Program

```
Correct Predictions:
9830400x4 Matrix{Bool}:
 0  1  1  0
 0  1  1  0
 0  1  1  0
 0  1  1  0
 ⋮
 1  1  0  0
 0  0  0  1
 1  1  0  0
 0  1  1  0
```

- Pembandingan predicted classes dengan class asli. Jika sama maka nilai = 1 (true), jika berbeda maka nilai = 0 (false)

```
Feature 1 Accuracy: 39.67047%
Feature 2 Accuracy: 36.10197%
Feature 3 Accuracy: 44.05227%
Feature 4 Accuracy: 40.380157%
```

- Akurasi dari prediksi berdasarkan euclidean distance dari tiap feature

```
New Averages:
3×4 Matrix{Float16}:
 -1.096   6.125  -2.451  -5.52
  4.516   1.264   3.059  -0.861
 10.55   -2.803   8.53    4.105
```

- Rata-rata baru diambil data yang hasil prediksinya benar

# 11. Evaluasi

```
Feature 1 Accuracy: 39.67047%
Feature 2 Accuracy: 36.10197%
Feature 3 Accuracy: 44.05227%
Feature 4 Accuracy: 40.380157%
```

Akurasi dari iterasi pertama menggunakan euclidean distance masih cukup rendah. Perlu dilakukan iterasi-iterasi selanjutnya dalam cascade classifier untuk memperbagus akurasi