PRA-UAS IEE2032 M13

Filipus Dani - 202000135

Ketentuan:

1. Seorang pengusaha berencana untuk mendirikan 3 Pabrik yang terkait satu sama lainnya
2. Ketiga Pabrik tersebut adalah
    1. Sistem Pertama = Smart Farm
        1. Subsistem1: Susu Hewani dan Telur
        2. Subsistem2: Daging Merah
        3. Subsistem3: Daging Putih
    2. Sistem Kedua = SmartPlantation
        1. Subsistem4: Sumber Karbohidrat: Beras, Gandum, Jagung
        2. Subsistem5: Sayuran
        3. Subsistem6: Buah-buahan
    3. Sistem Ketiga = SmartRestaurant
        1. Subsistem7: Deteksi Musim (Nov – Feb: Winter ➔ Daging Merah dibutuhkan, Summer ➔ Sayuran > Daging, Fall ➔ semua sama rata, Spring, Daging putih dibutuhkan). Random musim
        2. Subsistem8: Deteksi Hasil Penjualan Berfluktuasi. Random hasil penjualan
        3. Subsistem9: Deteksi jumlah pengunjung restoran. Random jumlah pengunjung

Sensor:

1. Sensor dihasilkan oleh Node-red
2. Data dari Sensor di kirim ke Django menggunakan MQTT
3. Tidak ada satu sensorpun yang sama fungsinya, baik di dalam satu subsistem maupun dalam satu system, maupun dalam ketiga pabrik. Setiap sensor adalah unik fungsinya
4. Kemungkinan besar pada subsistem memerlukan banyak macam sensor tetapi pilih 3 sensor saja yang paling utama. Jadi 3 sensor untuk setiap subsistem.
5. Setiap sensor memiliki jangkah (range) yang secara logis sesuai dengan fungsinya
    1. Misalkan sensor temperatur boiler, logisnya antara 200 – 500 derajat Celsius. Maka jangkah adalah 200-500, nilai random
    2. Jangka waktu update sensor diatur sendiri
6. Node-Red Dashboard menampilkan data setiap sensor dengan tata letak memperhatikan pengelompokkan sistem dan subsistem. Jadi sensor-sensor dikelompokkan dalam satu subsistem dan beberapa subsistem dikelompokkan dalam sistemnya

Desain Sensor

1. Pabrik Smart Farm
    1. Subsistem susu hewani dan telur
        a. Sensor Temperature Infrared (-50-380^{O}C)
        b. Sensor Conductivity (0-200 mS/cm)
        c. Sensor Flow Susu (0-10 L/min)
    2. Subsistem daging merah
        a. Sensor CO (0-500 ppm)
        b. Sensor Amonia (0-100 ppm)
        c. Sensor Pressure (0-500kPa)
    3. Subsistem daging putih
        a. Sensor O2 (0-25%)
        b. Sensor CO2 (0-5000 ppm)
        c. Sensor Water Level (0-10 m)

2. Pabrik Smart Plantation
    1. Subsistem sumber karbohidrat
        a. Sensor Soil Moisture (0-100%)
        b. Sensor pH (0-14 unit)
        c. Sensor Sunlight (0-1000 lux)
    2. Subsistem sayuran
        a. Sensor Humidity (0-100%)
        b. Sensor Anemometer (0-50 m/s)
        c. Sensor UV (0-1000 mW/cm^2)
    3. Subsistem buah-buahan
        a. Sensor Solar Radiation (0-2000 W/m^2)
        b. Sensor Berat Buah (0-10 kg)
        c. Sensor Color Hue (0-360^{O})

3. Pabrik Smart Restaurant
    1. Subsistem deteksi musim
        a. Sensor Barometer (0-200 kPa)
        b. Sensor Wind Vane (0-360^{O})
        c. Sensor Visibility (0-10 km)
    2. Subsistem deteksi hasil penjualan
        a. Sensor Inventory Stock (0-100%)
        b. Sensor Selling Rate (0-5000 item/day)
        c. Sensor Customer Satisfaction (0-10)
    3. Subsistem deteksi jumlah pengunjung restoran
        a. Sensor Occupancy (0-100%)
        b. Sensor Audio (0-90 dB)
        c. Sensor Sentimen (0-100%)

Desain Aktuator

1. Pabrik Smart Farm
    1. Subsistem susu hewani dan telur merah
        a. Aktuator Milking machine (0-100 mm/s)
    2. Subsistem daging merah
        a. Aktuator Slaughter machine (0-200 rad/s)
    3. Subsistem daging putih
        a. Aktuator Feeding machine (0-300 cm^2/s)
2. Pabrik Smart Plantation
    4. Subsistem sumber karbohidrat
        a. Aktuator Output gramasi karbohidrat (0-1000 gr/s)
    5. Subsistem sayuran
        a. Aktuator Sprayer (0-200 cm^3/s)
    6. Subsistem buah-buahan
        a. Aktuator Pemberi pupuk (0-5 kg)
3. Pabrik Smart Restaurant
    7. Subsistem deteksi musim
        a. Aktuator AC (16-32 ^OC)
    8. Subsistem deteksi hasil penjualan
        a. Aktuator Diskon (0-20%)
    9. Subsistem deteksi jumlah pengunjung restoran
        a. Aktuator Petunjuk Jumlah Kursi (0-100)

Aktuator smart farm : Output produk (0-1000 kg/hari)
Aktuator smart plantation : Hasil tanam (0-100 ton/bulan)
Aktuator Smart Restaurant : Proyeksi Keuntungan (-10-200 jt/bulan)

Aktuator Pabrik : Output nilai pabrik (0-100)

Sejarah pengerjaan

Apa saja yang sudah dikerjakan untuk tugas PraUAS sampai dengan saat ini?

- Membuat desain ketiga pabrik dengan masing-masing 3 subsistem dengan masing-masing 3 sensor dan 1 aktuator
- Membuat flow node-RED untuk menghasilkan data masing-masing sensor
- Mengirimkan dan menerima data sensor melalui MQTT
- Membuat aplikasi Django yang dapat menerima data sensor dari MQTT
- Menampilkan data sensor pada webview Django
- Mengimplementasi 9 algoritma Machine Learning Linear Regression untuk 9 sensor
- Mengimplementasi 4 algortima Machine Learning Linear Regression untuk masing-masing pabrik dan ketiga pabrik sekaligus

Apa kendala yang ditemukan?

- Menentukan 27 sensor yang berbeda, tetapi masih masuk akal untuk masing-masing subsistem dan pabrik. Kendala lainnya adalah membuat begitu banyak sensor. Setelah membuat beberapa sensor, tugas ini menjadi repetitif dan hanya mengulang membuat sensor, mqtt, dan dashboard pada node-RED.
- Repetisi membuat penerimaan 27 sensor pada Django
- Repetisi membuat 9 machine learning
- Repetisi membuat 4 machine learning

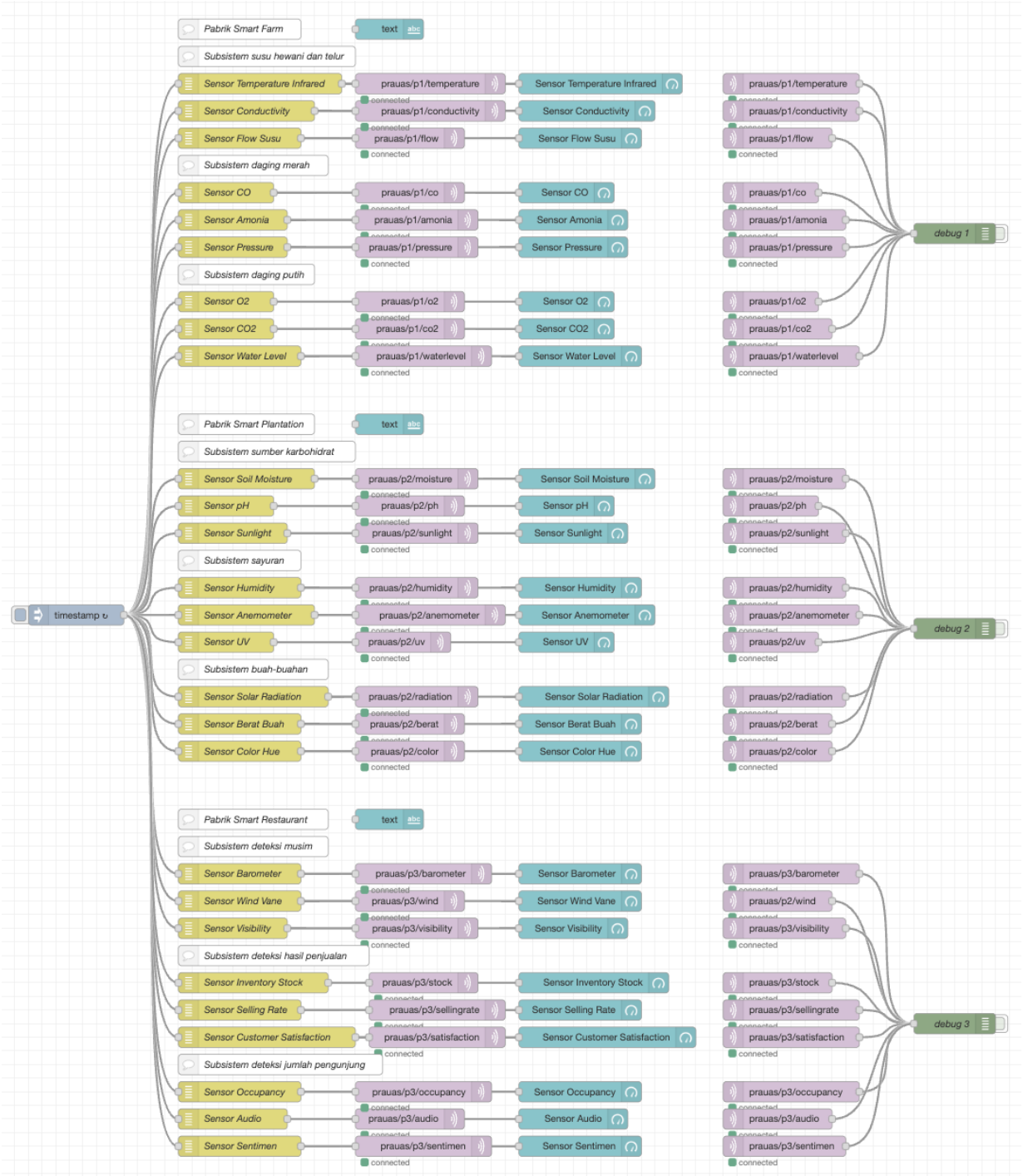
Solusi kendala yang ditemukan?

- Menentukan 27 sensor yang berbeda, tetapi masih masuk akal untuk masing-masing subsistem dan pabrik. Kendala lainnya adalah membuat begitu banyak sensor. Setelah membuat beberapa sensor, tugas ini menjadi repetitif dan hanya mengulang membuat sensor, mqtt, dan dashboard pada node-RED. – Solusi: Saya meminta bantuan pendapat teman-teman untuk bersama-sama berdiskusi sensor apa saja yang dapat digunakan. Untuk mengatasi masalah repetitif, saya hanya bisa bersabar dan terus membuat setiap sensor pada Node-RED
- Repetisi membuat penerimaan 27 sensor pada Django – Solusi: Menggunakan fitur multi cursor pada Visual Studio Code
- Repetisi membuat 9 machine learning – Solusi: Membuat skrip untuk mendapatkan data training
- Repetisi membuat 4 machine learning – Solusi: Membuat skrip untuk mendapatkan data training

Tautan Repo GITHUB

https://github.com/filipusdani/IEE3032-prauas

Flow Node-RED



Dashboard Node-RED

Dashboard Webview Django

Kode Machine Learning Linear Regression

```python
from sklearn import linear_model
import numpy as np

def hitung_milking_machine(a, b, c):
    X = np.array([
    [ -46.303, -2.3952222222222215, 46.2795555555555554, 96.32133333333334, 148.2341111111111, 201.88288888888889, 234.03466666666668, 308.54344444444445, 304.6222222222222, 399.351 ],
    [ 0.0, 22.34422222222222, 42.06044444444444, 70.79366666666665, 80.99888888888889, 104.61611111111111, 139.2313333333333, 150.49155555555555, 169.85677777777778, 195.598 ],
    [ 0.0, 1.0681111111111112, 2.1442222222222225, 3.2833333333333337, 4.520444444444444, 5.334555555555555, 6.096666666666667, 8.084777777777779, 9.011888888888889, 10.272 ]
    ])
    X = np.transpose(X)
    y = np.array([ 0.0, 12.17011111111111, 20.778222222222222, 31.297333333333327, 42.46044444444444, 57.21055555555556, 64.74666666666666, 71.87677777777778, 90.52288888888889, 90.589 ])
    reg = linear_model.LinearRegression().fit(X, y)
    return round((reg.predict([[a, b, c]]))[0],2)

def hitung_slaughter_machine(a, b, c):
    X = np.array([
    [ 0.0, 57.42155555555556, 114.41611111111112, 154.56366666666668, 229.28022222222222, 252.14277777777778, 345.3733333333334, 424.46988888888893, 429.74444444444447, 450.277 ],
    [ 0.0, 10.892111111111111, 22.528222222222222, 33.99133333333333, 47.49344444444444, 52.970555555555556, 66.93766666666666, 79.99377777777777, 85.23488888888889, 96.774 ],
    [ 0.0, 57.629555555555555, 120.80311111111112, 158.54666666666667, 233.30522222222223, 263.0727777777778, 315.24233333333336, 364.0388888888889, 471.98544444444445, 502.89 ]
    ])
    X = np.transpose(X)
    y = np.array([ 0.0, 21.07622222222222, 43.83244444444444, 67.26966666666665, 88.10188888888888, 109.58511111111112, 132.7303333333333, 160.95955555555554, 187.71777777777777, 216.872 ])
    reg = linear_model.LinearRegression().fit(X, y)
    return round((reg.predict([[a, b, c]]))[0],2)

def hitung_feeding_machine(a, b, c):
    X = np.array([
    [ 0.0, 2.8067777777777776, 5.760555555555555, 8.598333333333333, 10.74611111111111, 15.01588888888889, 16.163666666666664, 17.744444444444444, 22.33722222222222, 22.999 ],
    [ 0.0, 565.3955555555556, 1203.8291111111112, 1818.0386666666666, 2021.6362222222222, 2696.6967777777777, 3449.426333333333, 3648.110888888889, 4664.555444444444, 5382.644 ],
    [ 0.0, 1.1491111111111112, 2.146222222222223, 3.0543333333333336, 4.615444444444445, 5.711555555555555, 6.278666666666667, 8.2057777777778, 8.181888888888889, 10.229 ]
    ])
    X = np.transpose(X)
    y = np.array([ 0.0, 35.03333333333334, 61.35966666666667, 106.51, 138.85633333333334, 158.0436666666667, 185.421, 215.03233333333336, 253.5676666666667, 296.767 ])
    reg = linear_model.LinearRegression().fit(X, y)
    return round((reg.predict([[a, b, c]]))[0],2)

def hitung_gramasi_karbo(a, b, c):
    X = np.array([
    [ 0.0, 12.22011111111111, 20.947222222222223, 35.017333333333326, 46.74344444444444, 53.06755555555556, 64.09666666666666, 85.10477777777777, 80.69288888888889, 103.579 ],
    [ 0.0, 1.5895555555555556, 2.8941111111111111, 4.616666666666667, 5.657222222222222, 7.177777777777778, 9.175333333333334, 9.861888888888889, 13.679444444444444, 13.074 ],
    [ 0.0, 110.04111111111112, 207.62322222222224, 350.02433333333335, 417.75944444444445, 583.5865555555555, 726.7876666666667, 753.1227777777779, 941.9048888888889, 1034.361 ]
    ])
    X = np.transpose(X)
    y = np.array([ 0.0, 117.42511111111111, 201.44622222222222, 359.5003333333334, 488.7344444444445, 512.7585555555555, 693.8376666666668, 711.9337777777778, 819.4028888888889, 1028.621 ])
    reg = linear_model.LinearRegression().fit(X, y)
    return round((reg.predict([[a, b, c]]))[0],2)

def hitung_sprayer(a, b, c):
    X = np.array([
    [ 0.0, 10.680111111111112, 23.71622222222222, 34.23433333333333, 45.01244444444444, 59.031555555555556, 68.20266666666666, 71.68377777777778, 83.56588888888888, 97.518 ],
    [ 0.0, 5.6125555555555556, 11.491111111111111, 16.072666666666663, 22.40422222222222, 28.21477777777778, 35.612333333333325, 39.29988888888889, 47.04844444444444, 46.229 ],
    [ 0.0, 104.82211111111111, 226.15522222222222, 366.21833333333336, 444.2384444444444, 554.7495555555555, 620.9306666666668, 801.6977777777778, 841.5418888888889, 1069.834 ]
    ])
    X = np.transpose(X)
    y = np.array([ 0.0, 23.20422222222222, 43.65544444444444, 62.54166666666666, 97.07388888888889, 107.11111111111111, 134.8963333333333, 158.25055555555554, 186.47777777777776, 188.556 ])
    reg = linear_model.LinearRegression().fit(X, y)
    return round((reg.predict([[a, b, c]]))[0],2)
```

```python
def hitung_pemberi_pupuk(a, b, c):
    X = np.array([
    [ 0.0, 230.11422222222222, 461.91044444444447, 629.5916666666667, 806.2988888888889, 1156.6621111111111, 1227.0793333333336, 1651.7265555555557, 1727.0157777777779, 1978.575 ],
    [ 0.0, 1.0731111111111111, 2.183222222222222, 3.4733333333333336, 4.68744444444445, 5.513555555555555, 6.039666666666667, 7.142777777777779, 8.489888888888888, 9.518 ],
    [ 0.0, 40.296, 83.265, 116.597, 156.605, 202.428, 255.001, 302.762, 324.968, 324.793 ]
    ])
    X = np.transpose(X)
    y = np.array([ 0.0, 0.5245555555555556, 1.168111111111111, 1.7586666666666668, 2.4262222222222225, 2.912777777777775, 3.4323333333333337, 4.049888888888889, 4.718444444444445, 4.674 ])
    reg = linear_model.LinearRegression().fit(X, y)
    return round((reg.predict([[a, b, c]]))[0],2)

def hitung_aircon(a, b, c):
    X = np.array([
    [ 0.0, 22.862222222222222, 48.61544444444444, 69.52666666666666, 83.95188888888889, 115.38311111111112, 137.9163333333333, 161.23955555555554, 165.90977777777778, 192.934 ],
    [ 0.0, 42.067, 82.834, 118.511, 149.003, 183.512, 223.197, 274.499, 334.727, 359.976 ],
    [ 0.0, 1.0761111111111112, 2.099222222222222, 3.3233333333333337, 4.057444444444444, 5.442555555555555, 6.744666666666667, 8.20877777777778, 8.776888888888889, 9.462 ]
    ])
    X = np.transpose(X)
    y = np.array([ 17.089, 19.47877777777778, 20.169555555555558, 20.372333333333334, 24.72811111111111, 24.15388888888889, 27.894666666666666, 25.91144444444444, 30.51522222222222, 35.11 ])
    reg = linear_model.LinearRegression().fit(X, y)
    return round((reg.predict([[a, b, c]]))[0],2)

def hitung_hasil_penjualan(a, b, c):
    X = np.array([
    [ 0.0, 12.116111111111111, 20.30622222222222, 32.71833333333333, 42.56044444444444, 60.928555555555555, 67.25866666666666, 74.14277777777777, 81.80088888888889, 98.875 ],
    [ 0.0, 501.72155555555554, 1035.6861111111111, 1523.2266666666665, 2161.3122222222223, 2887.6407777777777, 3106.257333333333, 4097.87288888889, 4328.096444444444, 5170.778 ],
    [ 0.0, 1.1631111111111112, 2.2782222222222224, 3.5373333333333337, 4.487444444444445, 6.007555555555555, 6.116666666666667, 7.225777777777779, 8.62488888888889, 9.941 ]
    ])
    X = np.transpose(X)
    y = np.array([ 0.0, 2.2672222222222222, 4.366444444444444, 6.684666666666667, 8.866888888888889, 10.075111111111111, 12.051333333333334, 14.755555555555556, 16.60677777777778, 21.068 ])
    reg = linear_model.LinearRegression().fit(X, y)
    return round((reg.predict([[a, b, c]]))[0],2)

def hitung_jumlah_kursi(a, b, c):
    X = np.array([
    [ 0.0, 11.222111111111111, 22.03522222222222, 30.355333333333327, 47.855444444444444, 50.94755555555556, 69.68066666666665, 71.38077777777777, 84.09688888888888, 106.51 ],
    [ 0.0, 9.065, 18.383, 29.346, 37.879, 46.129, 62.278999999999996, 74.388, 75.179, 93.649 ],
    [ 0.0, 12.044111111111111, 21.983222222222222, 33.03933333333333, 46.78344444444444, 60.61955555555556, 64.42166666666665, 78.62177777777777, 95.83188888888888, 96.1 ]
    ])
    X = np.transpose(X)
    y = np.array([ 0.0, 10.92811111111111, 21.08222222222222, 31.011333333333333, 43.17044444444444, 55.480555555555554, 63.87866666666666, 83.69877777777778, 80.98388888888888, 105.56700000000001 ])
    reg = linear_model.LinearRegression().fit(X, y)
    return round((reg.predict([[a, b, c]]))[0],2)

def hitung_output_produk(a, b, c):
    X = np.array([
    [ 0.0, 10.985111111111111, 20.75322222222222, 36.377333333333326, 46.98544444444444, 53.81955555555556, 65.60666666666665, 84.96277777777777, 94.59588888888888, 93.989 ],
    [ 0.0, 20.67822222222222, 42.16244444444446, 61.38266666666666, 82.81888888888889, 117.25511111111112, 144.1283333333333, 166.25655555555554, 192.05877777777778, 199.695 ],
    [ 0.0, 32.31433333333334, 65.53166666666667, 90.654, 141.96333333333335, 178.5896666666667, 211.772, 229.13233333333335, 245.06666666666667, 317.196 ]
    ])
    X = np.transpose(X)
    y = np.array([ 0.0, 121.54311111111112, 217.58422222222222, 336.69233333333335, 454.9434444444445, 573.6125555555556, 625.9526666666668, 850.2547777777778, 959.829888888889, 1072.797 ])
    reg = linear_model.LinearRegression().fit(X, y)
    return round((reg.predict([[a, b, c]]))[0],2)
```

Referensi

- Dokumentasi Django: https://docs.djangoproject.com/en/4.2/
- Dokumentasi Node-RED: https://nodered.org/docs/
- Project Paho-MQTT: https://pypi.org/project/paho-mqtt/
- Dokumentasi Paho-MQTT: https://www.eclipse.org/paho/index.php?page=clients/python/docs/index.php
- Dokumentasi Mosquitto MQTT: https://mosquitto.org/documentation/