Anomaly Detection Model with Feature Engineering

1. **Pembuatan Feature Engineering**
   A. **Moving Average (MA)** - Deteksi Tren & Fluktuasi Stabil

   Moving Average (Rata-rata bergerak) merupakan salah satu metode untuk membantu model menghilangkan noise dan menangkap pola tren data. Moving Average dihitung menggunakan jendela waktu tertentu (slicing window) misalnya 3 jam, 20 menit, dll. Moving Average (MA) digunakan untuk menghaluskan fluktuasi jangka pendek dan mengidentifikasi tren jangka panjang dalam data time series.

   - Jika nilai sensor tiba-tiba naik/turun jauh dari MA, ini bisa menjadi anomali.
   - MA membantu mendeteksi perubahan tren yang tidak normal.

   Penghitungan :

$$MA_k(t) = \frac{X(t) + X(t-1) + ... + X(t-k+1)}{k}$$

   di mana:

   - $MA_k(t)$ = Moving Average pada waktu ke-t dengan jendela k.
   - $X(t)$ = Nilai sensor pada waktu t.
   - k = Ukuran jendela (misal 3 jam).

   B. **Rate of Change (ROC)/Derivative/Delta**- Deteksi Perubahan Cepat

   Rate of Change (ROC) mengukur seberapa cepat nilai sensor berubah dari waktu ke waktu. Ini sangat berguna untuk mendeteksi perubahan tiba-tiba yang bisa menjadi tanda anomali. ROC bias akita sebut sebagai delat perubahan suhu.

   - Jika ROC terlalu tinggi/terlalu rendah, berarti ada perubahan mendadak dalam data yang bisa menjadi anomali.
   - ROC negatif besar menunjukkan penurunan tajam, sedangkan ROC positif besar menunjukkan lonjakan tiba-tiba.

   Penghitungan :

$$ROC(t) = X(t) - X(t-1)$$

   di mana:

   - $ROC(t)$ = Perubahan nilai sensor pada waktu t.
   - $X(t)$ = Nilai sensor pada waktu t.
   - $X(t-1)$ = Nilai sensor sebelumnya.

   C. **Rolling Standard Deviation –** Deteksi Volatilitas

   Rolling Standard Deviation (Rolling Std Dev) mengukur seberapa bervariasi nilai sensor dalam jendela waktu tertentu. Jika data menjadi terlalu fluktuatif atau terlalu stabil secara tiba-tiba, ini bisa menjadi indikasi anomali.

   - Jika std dev tinggi, berarti data menjadi lebih tidak stabil, bisa jadi ada gangguan pada sistem.
   - Jika std dev sangat rendah tiba-tiba, bisa jadi ada kesalahan pada sensor (sensor stuck).

   Penghitungan :

$$\text{Rolling Std}_k(t) = \sqrt{\frac{\sum_{i=t-k+1}^{t}(X(i) - \bar{X})^2}{k}}$$

   di mana:

   - $\bar{X}$ = Rata-rata dalam jendela k.
   - $X(i)$ = Nilai sensor pada waktu i.

Dalam pendeteksian anomaly menggunakan fitur rolling standari deviasi, sebuah titik data dikatakan anomali jika nilainya jauh dari nilai **threshold** (rata-rata, biasanya lebih dari 2 atau 3 kali standar deviasi)

$$\text{Upper Threshold} = \bar{X} + 2 \cdot \sigma$$

$$\text{Lower Threshold} = \bar{X} - 2 \cdot \sigma$$

di mana:

- $\bar{X}$ = **Rata-rata (Mean)** dalam jendela waktu tertentu.
- σ = **Standard Deviation (Std Dev)** dalam jendela waktu tertentu.

D. Fast Fourier Transform (FFT)

Fast Fourier Transform (FFT) adalah algoritma yang digunakan untuk mengubah data dari domain waktu ke domain frekuensi. Dalam konteks anomaly detection pada data sensor, FFT digunakan untuk mengekstrak fitur dari pola frekuensi dalam data sensor, sehingga memungkinkan pendeteksian perubahan signifikan atau anomali yang tidak terlihat dalam domain waktu

Peran FFT dalam Data Sensor

Data dari sensor sering kali berupa **deret waktu** (time series) yang mencatat nilai dari suatu parameter (misalnya suhu, tekanan, getaran) pada interval waktu tertentu. Dengan menggunakan FFT, kita bisa:

- **Mengidentifikasi pola normal** dalam data berdasarkan komponen frekuensinya.
- **Mendeteksi anomali** sebagai sinyal dengan frekuensi yang tidak biasa atau perubahan drastis dalam spektrum frekuensi.
- **Menghilangkan noise** dengan menyaring frekuensi yang tidak relevan sebelum analisis lebih lanjut.

Formula dasar FFT:

$$X(f) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi fn/N}$$

- $x(n)$ adalah data suhu
- N adalah jumlah sampel (600 dalam kasus ini)
- $e^{-j2\pi fn/N}$ adalah transformasi Fourier

Hasil FFT akan memberikan **frekuensi dominan** dalam data suhu.

Dalam Bahasa python, fungsi FFT ini sudah ada dalam modul ***scipy.fftpack***, dimana penggunannya tinggal memanggil ***fft(sensor_value).*** Hasil dari FFT ini merupakan bilangan kompleks, sehingga harus melakukan perhitungan lain untuk dijadikan input sebuha model, yaitu menghitung ***Magnitude***(Amplitudo) dan ***Frequency*** dari FFT. Untuk analisis sinyal nyata, hasil spektrum dari Magnitude dan Frequency biasanya akan diambil **setengahnya, karena frekuensi negative adalah ceminan dari frekuensi positif.** Dari fitur Magnitude dan Frequency yang dihasilkan, biasanya tidak langsung dijadikan input di pemodelan machine learning. Fitur tersebut akan diekstrak menjadi fitur-fitur penting sehingga modelnya lebih efisien.

- Fitur yang diekstrak dari Magnitude : Nilai puncak frekuensi dominan (peak magnitude), Frekuensi dominan (peak frequency), Total energi spektrum (sum of magnitudes), Mean dan standard deviation dari magnitudo FFT

- Fitur yang diekstrak dari Frekuensi : Frekuensi rata-rata (weighted mean frequency) dan Spread spektrum (bandwidth)

Analisis lanjutan ekstraksi fitur menjadi input model Anomaly Detection

✓ **Tanpa roling window**
Ketika fitur frekuensi dan magnitude diekstrak menjadi fitur-fitur sebelumnya ke keseluruhan dataset sekaligus, maka value yang akan dihasilkan akan 1 nilai (1 row saja). Metode ini hanya cocok untuk melihat pola secara general. **Tidak cocok untuk pendeteksian early warning**

✓ **Dengan rolling window**
Penggunaan rolling window akan memudahkan untuk early warning, karena kita akan mendefinisikan window size untuk setiap berapa titik data yang akan dijadikan window untuk penghitungan ektraksi fitur. Misalkan window size = 100 titik suhu, maka setiap 100 titik suhu akan dikonversi menjadi 1 baris fitur untuk menghitung ekstraksi fitur yang akan dijadikan input model. Jika memiliki 600 baris data , jumlah window akan dibuat sekitar :

$$\frac{600 - 100}{10} + 1 = 51 \text{ baris}$$

Penentuan **window size** juga harus mempertimbangkan pola yang ingin ditagkap dari sebuah data dan syarat jumlah data yang harus terpenuhi ketika akan dilanjutkan dengan analisis machine learning. Metode ini lebih cocok untuk pendeteksian anomaly setiap segmen waktu dan sebagai early warning anomaly detection.

2. Model

A. Unsupervised Learning

- **Isolation Forest**

Isolation Forest (IF) adalah algoritma berbasis pohon (tree-based algorithm) yang dikembangkan khusus untuk deteksi anomali. **Prinsip dasarnya adalah bahwa anomali lebih mudah diisolasi dibandingkan dengan data normal**.

✓ Data anomali berada jauh dari distribusi utama, sehingga lebih cepat terpisah dalam struktur pohon.

✓ Algoritma ini bekerja dengan cara membangun beberapa pohon keputusan (isolation trees) dan mengukur rata-rata kedalaman yang dibutuhkan untuk mengisolasi suatu sampel.

✓ Semakin rendah kedalaman suatu titik data terisolasi, semakin besar kemungkinan data tersebut adalah anomali.

Dalam model Isolation Forest score anomaly dihitung dengan rumus

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$$

- $E(h(x))$ = Rata-rata panjang jalur dari beberapa isolation trees.
- $c(n)$ = Fungsi koreksi berdasarkan ukuran sampel.

Ketika skor anomaly mendekati 0, maka data dianggap normal. Namun, ketika skor anomaly menedekati 1 maka data dianggap anomaly. Dalam model ini parameter yang biasa diperhatikan yaitu parameter **contamination,** parameter ini berarti bahwa kitab isa mendefinisikan seberapa banyak data yang dapat diharapkan sebagai anomaly. Misalnya 0.1, 0.01, 0.5, 0.15, dst. Pendefinisian parameter ini berguna ketika dari tim teknisi ada standar acuan dari history sebelumnya terkait batas anomaly yang sering terjadi.

- **One Class SVM**

  One-Class SVM (OCSVM) adalah algoritma berbasis Support Vector Machine (SVM) yang digunakan untuk deteksi anomali dalam skenario unsupervised learning. Prinsip utamanya adalah mencari hyperplane yang memisahkan data normal dari daerah kosong yang tidak mengandung data.

  ✓ Model dilatih hanya dengan data normal dan bertujuan untuk membentuk boundary yang memisahkan data normal dari daerah anomali.
  ✓ Data yang berada di luar boundary ini dianggap sebagai anomali.

  Dalammodel One Class SVM penentuan boundary dengan penghitungan **margin** yang membedakan antara data normal dan anomali. Parameter yang penting dalam model ini yaitu **nu** (menentukan persentase data yang dianggap sebagai outlier dan **gamma** (Mengontrol jarak pengaruh titik data dalam ruang fitur). Jika suatu titik data itu mendekati sebuah boundary yang ditentukan maka data dianggap normal, sebaliknya jika data menjauhi boundary (di luar boundary) maka data dianggap sebagai anomaly.

  Penentuan anomaly juga bisa dihitung dari nilai decision function dari model. Nilai ini dihitung berdasarkan jarak dari hyperplane. Nilai decision score ini bisa dibandingkan dengan threshold yang kita definisikan.

B. Supervised Learning dengan Metode Lazy Predict

Dalam pengujian model supervised learning, pengujian bisa dilakukan secara efisien menggunakan library yang ada di Python yaitu **LazyPredict.** LazyPredict memungkinkan pengguna untuk mencoba beberapa algoritma machine learning dengan sedikit konfigurasi, sehingga pengguna dapat memilih model yang memberikan hasil terbaik dari berbagai model yang tersedia.

LazyPredict memudahkan dalam menguji model-model yang berbeda, tanpa memerlukan pengetahuan yang mendalam tentang parameter tuning atau pemilihan algoritma. Secara garis besar, LazyPredict dapat mempercepat proses eksperimen dan membantu menemukan model yang cocok untuk masalah yang dihadapi.

Dalam **supervised learning**, model dilatih menggunakan data yang sudah diberi label. Model akan mencoba untuk mempelajari hubungan antara **fitur (independent variables)** dan **target (dependent variable)**. Setelah itu, model dapat digunakan untuk **memprediksi label atau output** berdasarkan data yang tidak terlihat sebelumnya. LazyPredict, sebagai metode yang memungkinkan eksperimen dengan berbagai algoritma, dapat digunakan untuk berbagai jenis masalah supervised learning seperti Regresi dan Klasifikasi. Dalam case ini, model akan dibuat target biner yaitu 0 dan 1, sehingga Lazy Predict yang digunakan yaitu untuk klasifikasi.

LazyPredict bekerja dengan cara otomatis menguji sejumlah besar model supervised learning pada dataset yang Anda berikan. Model-model ini kemudian akan dievaluasi menggunakan **cross-validation** untuk memberikan hasil yang paling representatif mengenai performa model pada data yang tidak terlihat. Dalam pemodelan menggunakan LazyPredict dataset yang akan diuji harus sudah dilakukan pre-processing data dan juga sudah dibagi menjadi train dan test data (fitur X dan y).

```
X = data.data
y = data.target

# Membagi dataset menjadi data pelatihan dan data pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Setelah itu, data bisa langsung diuji menggunakan LazyClassfier untuk mendapatkan rekomendasi model dengan metric evaluasinya masing-masing

```
from lazypredict.Supervised import LazyClassifier

from sklearn.model_selection import train_test_split

clf = LazyClassifier(verbose=0,ignore_warnings=True, custom_metric=None)
models, predictions = clf.fit(X_train, X_test, y_train, y_test)
model_dictionary = clf.provide_models(X_train,X_test,y_train,y_test)
models
```

Contoh hasil output dari LazyClassifier

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
NearestCentroid	0.60	0.61	0.61	0.63	0.12
PassiveAggressiveClassifier	0.44	0.59	0.59	0.44	0.01
AdaBoostClassifier	0.75	0.50	0.50	0.65	0.16
LinearDiscriminantAnalysis	0.75	0.50	0.50	0.65	0.03
SVC	0.75	0.50	0.50	0.65	0.02
SGDClassifier	0.75	0.50	0.50	0.65	0.02
RidgeClassifierCV	0.75	0.50	0.50	0.65	0.03
RidgeClassifier	0.75	0.50	0.50	0.65	0.02
QuadraticDiscriminantAnalysis	0.75	0.50	0.50	0.65	0.02
Perceptron	0.75	0.50	0.50	0.65	0.03
LogisticRegression	0.75	0.50	0.50	0.65	0.03
LinearSVC	0.75	0.50	0.50	0.65	0.02
LabelSpreading	0.75	0.50	0.50	0.65	0.04
LabelPropagation	0.75	0.50	0.50	0.65	0.03

Dari rekomendasi model yang doberikan LazyPredict bisa menjadi pertimbangan atau sebagai baseline model seblum melakukan tuning lebih lanjut. LazyPredict hanya menerapkan **default parameter** sehingga analisis lebih lanjut untuk peningkatan akurasi bisa dilakukan dengan hyperparamter tuning. Tetapi LazyPredict ini sangat membantu dalam pengujian banyak model dalam waktu singkat dan otomatis.

REFERENSI SCRIPT

1. Script Pemodelan FFT : experiment_FFT.ipynb
2. Script Pemodelan Supervised Learning wit MA, ROC, STD : experiment_Supervised_Learning.ipynb
3. Script Pemodelan Unsupervised Learning with MA, ROC, STD: experiment_Unsupervised_Learning.ipynb