ANOMALY DETECTION ON SOFTWARE LICENSE DATA USING MACHINE LEARNING MODELS AT PERTAMINA GEOTHERMAL ENERGY TBK

NAME	KEIKO PRAKOSO
FACULTY	COMPUTER SCIENCE (ARTIFICIAL INTELLIGENCE)
MATRIC NUMBER	S2139335
COMPANY	PERTAMINA GEOTHERMAL ENERGY TBK
COMPANY SUPERVISOR	RYAN FADILLAH
SUPERVISOR EMAIL	ryan.fadillah@pertamina.com
JOB TITLE	Assistant Manager IT Operation
FACULTY SUPERVISOR	DR. TUTUT HERAWAN

1. Latar Belakang

Dalam lingkungan kerja enterprise yang kompleks, penggunaan software berlisensi secara legal menjadi salah satu aspek penting dalam menjaga kepatuhan terhadap regulasi dan keamanan sistem informasi. Namun, dalam praktiknya, sering kali ditemukan penggunaan software tidak resmi (bajakan) atau software yang tidak lagi memiliki lisensi aktif, baik secara sengaja maupun tidak disadari oleh pengguna. Hal ini tidak hanya meningkatkan risiko terhadap keamanan data perusahaan, tetapi juga dapat berdampak pada aspek hukum dan reputasi organisasi.

Proses identifikasi software yang mencurigakan secara manual memakan waktu, tidak efisien, dan sangat bergantung pada pengecekan satu per satu yang tidak scalable ketika jumlah perangkat dan software sangat banyak. Oleh karena itu, dibutuhkan sistem otomatis yang mampu mendeteksi software-software yang berpotensi anomali atau tidak wajar secara cepat dan akurat.

Melalui pemanfaatan teknologi machine learning, terutama pendekatan *unsupervised anomaly detection*, proses ini dapat diotomatisasi dengan lebih efisien. Dalam proyek ini, dilakukan penerapan beberapa model machine learning, yaitu Isolation Forest, One-Class SVM, Local Outlier Factor (LOF), dan Autoencoder, untuk mendeteksi pola software yang tidak umum dari dataset yang telah disediakan oleh pihak perusahaan.

Proyek ini diharapkan dapat menjadi pondasi awal dalam mengembangkan sistem pendeteksian anomali software yang adaptif dan scalable, serta membantu tim IT dalam mengambil keputusan berbasis data terhadap penggunaan software di dalam organisasi.

2. Tujuan Proyek

Tujuan dari proyek ini adalah untuk mengembangkan dan mengevaluasi sistem deteksi anomali pada penggunaan software di lingkungan perusahaan menggunakan pendekatan machine learning. Sistem ini dirancang untuk membantu tim IT dalam mengidentifikasi software yang mencurigakan, tidak umum, atau berpotensi tidak memiliki lisensi resmi tanpa perlu pengecekan manual satu per satu.

Secara khusus, proyek ini bertujuan untuk:

1. Mengolah dan membersihkan data software yang berasal dari hasil inventarisasi perusahaan.

2. Melakukan rekayasa fitur (feature engineering) untuk menyiapkan data agar dapat digunakan dalam proses training model.

3. Menerapkan dan membandingkan beberapa model deteksi anomali, seperti Isolation Forest, One-Class SVM, Local Outlier Factor, dan Autoencoder, untuk mendeteksi pola yang tidak wajar dalam penggunaan software.

4. Mengevaluasi hasil deteksi anomali dari masing-masing model dan mencari kesamaan software yang dideteksi mencurigakan oleh beberapa model sekaligus.

5. Menyediakan visualisasi dan pelaporan hasil deteksi dalam format yang dapat dibaca oleh tim pengambil keputusan non-teknis.

6. Memberikan rekomendasi model yang paling efisien dan akurat untuk digunakan di masa depan dalam pengawasan penggunaan software di lingkungan perusahaan.

3. Ruang Lingkup Proyek

Ruang lingkup proyek ini mencakup seluruh tahapan dalam proses deteksi anomali software menggunakan pendekatan machine learning, mulai dari pengumpulan data hingga penyajian hasil. Adapun ruang lingkup proyek ini dibatasi pada poin-poin berikut:

3.1. Sumber Data

Dataset yang digunakan dalam proyek ini merupakan hasil rekapitulasi data perangkat lunak yang diinstal pada komputer-komputer di lingkungan kerja PT Pertamina Geothermal Energy. Data tersebut dikumpulkan dari internal monitoring tools dan sistem pencatatan software inventory yang telah tersedia sebelumnya.

Dataset utama yang dianalisis dalam proyek ini memiliki ukuran **4092 baris dan 8 kolom**, yang masing-masing berisi informasi sebagai berikut:

5CG2430TFG-LMB

Computer Name	Last Logged On User	Publisher	Product Name	Product Key	License Code	License Version	Install Date
5CG2430TFG-LMB	mk.bayu.wismono	Microsoft	Internet Explorer	9NDMK-Q8XTQ-HR93X-JD6HP-2DK86	00355-63081-87825-AAOEM		
5CG2430TFG-LMB	mk.bayu.wismono	Microsoft	Windows 10 Pro	9NDMK-Q8XTQ-HR93X-JD6HP-2DK86	00355-63081-87825-AAOEM		
5CG2430TFG-LMB	mk.bayu.wismono	Microsoft Corporation	Kaspersky Endpoint Security for Windows	*****-*****-*****-*****-WFG99	12345-679-1111111-30252	11.11.0.452	20240222

5CG2430TFK-LMB

Computer Name	Last Logged On User	Publisher	Product Name	Product Key	License Code	License Version	Install Date
5CG2430TFK-LMB	maulidyah.pratiwi	Microsoft	Internet Explorer	DVM3W-RN32K-PBGK6-3VWBQ-T6P2T	00355-63081-61876-AAOEM		
5CG2430TFK-LMB	maulidyah.pratiwi	Microsoft	Windows 10 Pro	DVM3W-RN32K-PBGK6-3VWBQ-T6P2T	00355-63081-61876-AAOEM		
5CG2430TFK-LMB	maulidyah.pratiwi	Microsoft Corporation	Kaspersky Endpoint Security for Windows	*****-*****-*****-*****-WFG99	12345-679-1111111-56708	11.8.0.384	20240516

Kolom	Deskripsi
Computer_Name	Nama unik komputer pengguna
Last_Logged_User	Nama pengguna terakhir yang login
Publisher	Nama penerbit perangkat lunak
Product_Name	Nama perangkat lunak yang terinstal
Product_Key	Nomor lisensi perangkat lunak
License_Code	Kode lisensi yang didaftarkan
License_Version	Versi lisensi perangkat lunak
Install_Date	Tanggal instalasi perangkat lunak

Data ini diperoleh dalam format CSV, dan menjadi dasar dari seluruh proses analisis anomali dalam proyek ini.

3.2. Proses Preprocessing Data

Pra-pemrosesan data dilakukan untuk memastikan bahwa data bersih dan dapat digunakan oleh model machine learning secara optimal. Proses ini meliputi beberapa tahapan penting sebagai berikut:

3.2.1. Deskripsi Dataset Awal

Dataset yang digunakan adalah Software_License_Cleaned.csv, dengan karakteristik sebagai berikut:

Jumlah entri (baris): 4.092 dan jumlah fitur (kolom): 8

- Computer_Name
- Last_Logged_User
- Publisher
- Product_Name
- Product_Key
- License_Code
- License_Version
- Install_Date

3.2.2. Pembersihan Data

Beberapa langkah pembersihan data dilakukan:

- Penghapusan entri duplikat untuk menghindari bias dari software yang terinstal ganda.
- Penanganan missing values:
    - Kolom kritis (Product_Name, Publisher, License_Code) digunakan hanya jika tidak kosong.
    - Kolom non-kritis seperti Product_Key, License_Version, dan Install_Date dibiarkan kosong atau diisi dengan

placeholder karena tidak berpengaruh langsung terhadap proses deteksi anomali.

3.2.3. Penghapusan Software Khusus

Beberapa entri software dikeluarkan dari dataset karena bukan merupakan anomali atau bersifat default/online:

- Microsoft 365 Apps for enterprise → menggunakan lisensi online berbasis subscription.
- Cisco AnyConnect Secure Mobility Client → tool resmi koneksi VPN.
- Internet Explorer → software default bawaan sistem operasi.

3.2.4. Enkoding Kategorikal

Model machine learning tidak dapat menerima input dalam bentuk teks. Oleh karena itu, dilakukan proses Label Encoding untuk mengubah kolom kategorikal:

- Product_Name
- Publisher
- License_Code

Setiap nilai unik diubah menjadi angka yang mewakili kategorinya.

Contoh:

- Microsoft → 1
- Kaspersky → 2

3.2.5. Normalisasi

Untuk menjaga stabilitas model terutama pada deep learning (Autoencoder), dilakukan proses Min-Max Normalization menggunakan MinMaxScaler agar semua nilai berada dalam rentang [0.0, 1.0].

Contoh:

- Jika Publisher memiliki rentang awal 0–50, maka nilai tersebut diubah menjadi nilai relatif dari 0.0 hingga 1.0.
- Normalisasi ini mencegah satu fitur mendominasi fitur lain selama proses pelatihan model.

3.2.6. Rasionalisasi Fitur (Feature Selection)

Setelah proses pembersihan, dipilih tiga fitur utama untuk digunakan dalam model deteksi anomali:

- Product_Name
- Publisher
- License_Code

Alasan pemilihan:

- Ketiga fitur ini paling mewakili identitas dan legalitas dari software.
- Fitur lainnya seperti Install_Date, License_Version, dan Product_Key memiliki banyak missing values atau variasi yang rendah secara statistik.
- Fitur yang dipilih memiliki korelasi yang lebih kuat dalam membedakan pola software legal dan ilegal.

3.3. Model yang Digunakan

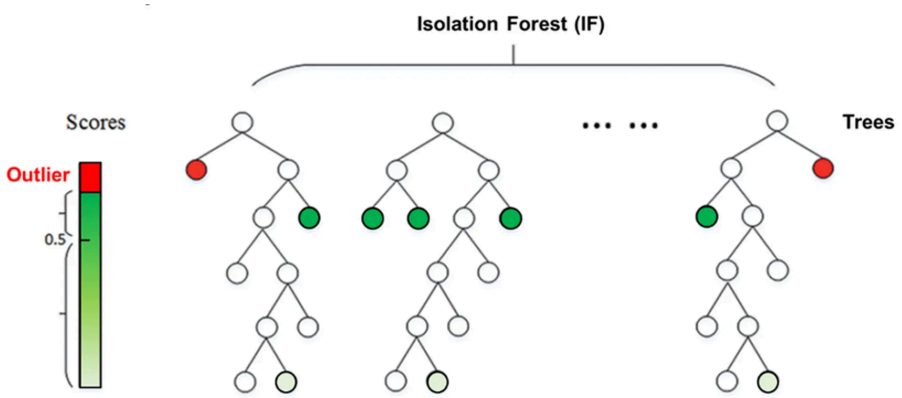
Untuk mendeteksi anomali pada data software yang digunakan di perusahaan, proyek ini mengimplementasikan empat metode deteksi anomali berbasis unsupervised learning. Pemilihan metode dilakukan untuk membandingkan efektivitas tiap pendekatan serta meningkatkan keakuratan melalui validasi silang antar model. Berikut adalah ringkasan keempat model yang digunakan:

- **Unsupervised Learning Models**:

  3.3.1. Isolation Forest (IF)

  Isolation Forest adalah algoritma berbasis pohon yang efektif dalam mendeteksi anomali dengan prinsip isolasi data. Algoritma ini bekerja dengan membangun sejumlah pohon dan secara acak memotong data untuk mengisolasi anomali, yang umumnya dapat dipisahkan dengan sedikit potongan.

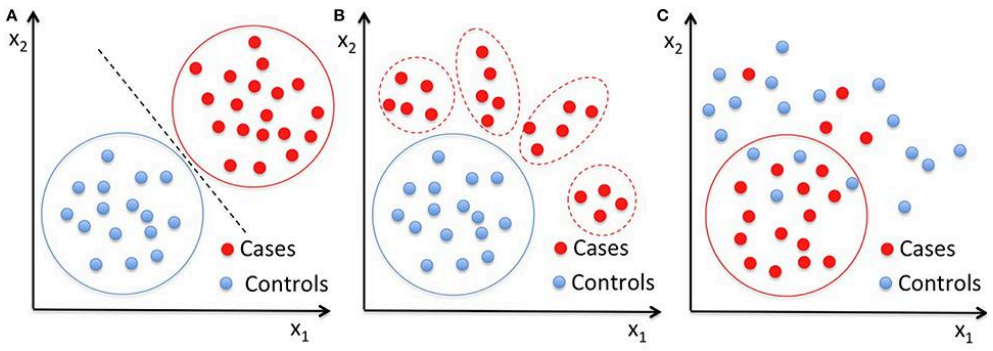
  - Dapat menangani jumlah besar software secara efisien.
  - Tidak memerlukan asumsi distribusi data tertentu, ideal untuk data software dengan keragaman tinggi.
  - Dapat digunakan sebagai baseline model untuk deteksi awal software yang memiliki pola distribusi berbeda dari mayoritas (misalnya software trial atau yang jarang digunakan).



3.3.2. One-Class Support Vector Machine (SVM)

One-Class SVM bekerja dengan membangun batas (hyperplane) yang memisahkan data mayoritas (normal) dari outlier berdasarkan margin maksimum.

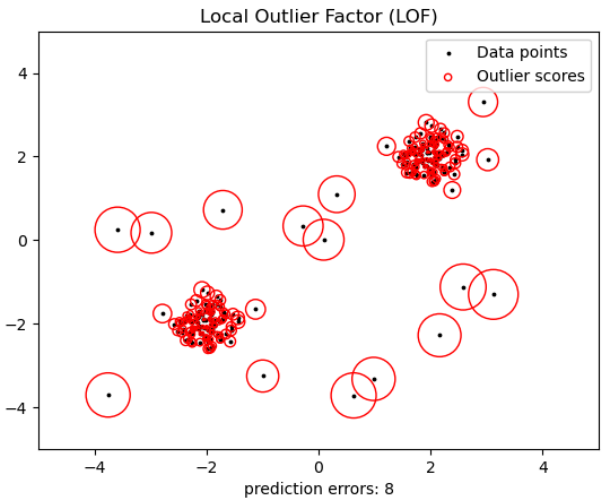
- Cocok digunakan untuk validasi anomali dalam dataset kecil hingga menengah, di mana pola-pola "normal" cukup jelas.
- Membantu menangkap software yang secara matematis berada jauh dari distribusi umum, misalnya software yang hanya diinstal oleh satu user atau tidak memiliki metadata yang lengkap.
- Digunakan sebagai pembanding terhadap hasil model IF dan Autoencoder yang cenderung lebih fleksibel.



3.3.3. Local Outlier Factor (LOF)

LOF menghitung tingkat keanehan suatu data berdasarkan kerapatan tetangganya. Model ini cocok digunakan untuk mendeteksi anomali lokal dalam kelompok data yang padat. LOF sangat berguna sebagai validasi silang terhadap hasil model IF atau Autoencoder karena mampu mendeteksi anomali yang hanya tampak "aneh" dibanding tetangganya, bukan seluruh data.

- Sangat bermanfaat untuk mendeteksi anomali lokal seperti software freeware atau open source yang hanya muncul di beberapa mesin spesifik.
- Digunakan sebagai metode validasi silang terhadap hasil dari Isolation Forest dan Autoencoder.

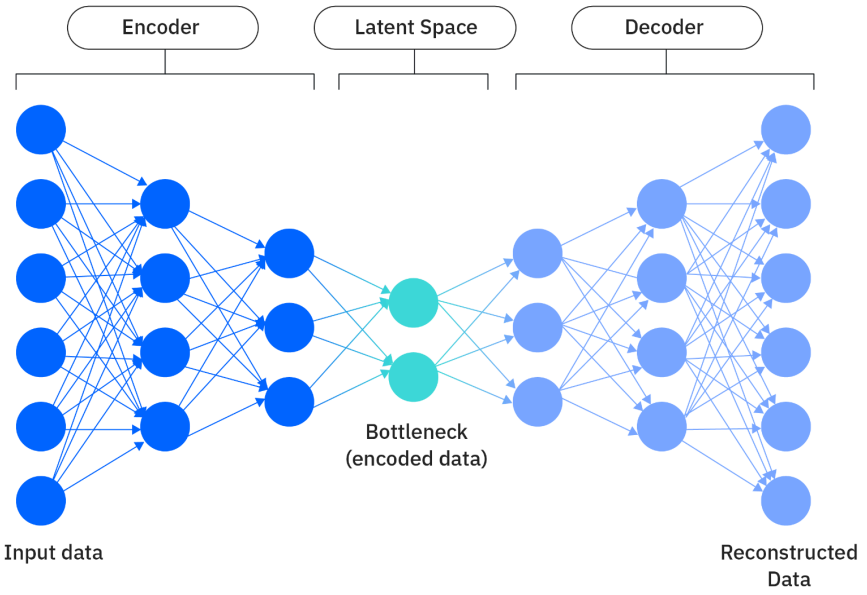


3.3.4. Autoencoder

Autoencoder adalah model neural network yang dilatih untuk merekonstruksi input-nya sendiri. Ketika input

adalah data normal, hasil rekonstruksi akan mirip. Namun, jika data tersebut anomali, error rekonstruksi akan tinggi.

- Cocok untuk menangkap relasi kompleks antar fitur software (misalnya antara Publisher dan License_Code).
- Mampu mendeteksi anomali non-linear yang sulit ditangkap oleh model tradisional.
- Memberikan pendekatan lebih mendalam (deep learning) dibanding model klasik, serta menjadi opsi yang baik untuk dievaluasi lebih lanjut dan dikembangkan sebagai sistem otomatis.



3.4. Analisis & Visualisasi

- Visualisasi tren dan distribusi software anomali.

- Visualisasi hasil kombinasi model (software yang terdeteksi oleh ≥ 3 model).

- Pemetaan kembali nama software dari hasil prediksi agar dapat dipahami oleh pengguna non-teknis.

3.5. Batasan

- Model tidak mengevaluasi legalitas software secara langsung, hanya berdasarkan pola perilaku data.

- Integrasi API eksternal seperti Kaseya belum diterapkan dalam laporan akhir ini dan disiapkan sebagai opsi lanjutan.

4. Metodologi

4.1. Alur Umum Proyek

Proyek ini mengadopsi pendekatan unsupervised learning untuk mendeteksi perangkat lunak mencurigakan dari data lisensi perusahaan. Tidak adanya label "benar atau salah" dalam data membuat pemilihan metode dan desain pipeline menjadi tantangan tersendiri, terutama dalam membangun sistem yang dapat meniru intuisi manusia secara otomatis.

4.1.1. Pengumpulan Data

Data lisensi perangkat lunak diperoleh langsung dari lingkungan operasional PT Pertamina Geothermal Energy dalam format .csv. Data ini

merupakan representasi riil dari software yang digunakan dalam infrastruktur IT, yang mencakup:

- Identitas perangkat (Computer_Name)
- Pengguna terakhir (Last_Logged_User)
- Nama produk, penerbit, dan kode lisensi
- Tanggal instalasi dan versi lisensi

4.1.2. Preprocessing Data

1. Deskripsi Dataset Awal:

- Sumber: File Software_License_Cleaned.csv
- Jumlah Baris: 4.092 data entri software
- Jumlah Kolom: 8 fitur utama
    - Computer_Name
    - Last_Logged_User
    - Publisher
    - Product_Name
    - Product_Key
    - License_Code
    - License_Version
    - Install_Date

2. Masalah Data:

- Missing Values:
    - License_Version: 2.626 missing
    - Install_Date: 3.357 missing
    - Product_Key: 1.484 missing

    - Kolom lainnya seperti Publisher dan Product_Name juga punya lebih dari 700 missing values.
- Duplikasi: Ada beberapa data software yang terdeteksi terduplikasi antar perangkat atau terinstall lebih dari sekali.

3. Penanganan Missing Value dan Duplikasi:

- Baris dengan missing value di fitur kritikal seperti Product_Name, Publisher, dan License_Code dihapus untuk menjaga kualitas data.
- Beberapa fitur seperti License_Version dan Install_Date diisi dengan nilai default (contoh: 0 untuk numerik, atau median untuk kolom Days_Since_Install).
- Data dikelompokkan (grouped) berdasarkan Product_Name untuk menghindari pengulangan deteksi pada entri software yang sama namun berasal dari komputer berbeda.

4. Normalisasi dan Encoding:

- Kolom Product_Name, Publisher, dan License_Code diubah menjadi numerik menggunakan LabelEncoder.
- Fitur numerik seperti License_Version dan Days_Since_Install dinormalisasi menggunakan MinMaxScaler ke skala 0-1.

5. Feature Selection (Rasionalisasi Fitur)
Dipilih tiga fitur utama: Product_Name, Publisher, dan

License_Code. Pemilihan ini berdasarkan pertimbangan bahwa fitur-fitur ini merupakan identitas utama dari perangkat lunak dan memiliki korelasi tinggi terhadap potensi anomali lisensi.

6. Train-Test Split

Dataset dibagi menjadi data latih dan data uji (80:20) untuk memungkinkan evaluasi performa model secara objektif.

4.2. Alur Proses Deteksi Anomali Software (Flowchart)

No	Tahap Proses	Penjelasan
1	Load Dataset	Membaca data dari file CSV yang berisi informasi software yang terinstal.
2	Preprocessing	Menangani nilai kosong, formatting tanggal, dan menghapus duplikat.
3	Feature Engineering	Memilih fitur penting seperti Product_Name, Publisher, License_Code, lalu melakukan encoding dan normalisasi.
4	Train-Test Split	Membagi data menjadi data latih dan data uji.
5	Model Training (4 Model)	Melatih empat model: IF, SVM, LOF, dan Autoencoder secara terpisah.
6	Anomaly Detection	Masing-masing model mendeteksi software yang tidak biasa (anomali).
7	Combine & Filter Anomaly	Hasil prediksi digabung, lalu disaring software yang terdeteksi oleh ≥3 model.
8	Visualisasi & Ekspor	Menampilkan hasil melalui grafik dan menyimpan hasil ke file CSV.
9	Pelaporan	Menyusun laporan dan menganalisis hasil deteksi anomali.

4.3. Pemilihan Model Deteksi Anomali

Model yang digunakan memiliki pendekatan *unsupervised learning*:

4.3.1. Isolation Forest (IF)

memisahkan anomali berdasarkan struktur pohon isolasi.

```
from sklearn.ensemble import IsolationForest

from sklearn.preprocessing import LabelEncoder


# Load data yang sudah diproses

df =
pd.read_csv("Software_License_Preprocessed.csv")


# Pilih fitur yang akan digunakan

fitur_terpilih = ["Product_Name", "Publisher",
"License_Code"]  # Pastikan semua ada di dataset

df_filtered = df[fitur_terpilih].copy()


# Encode fitur kategorikal menjadi numerik

encoder = LabelEncoder()

for col in ["Product_Name", "Publisher",
"License_Code"]:

    df_filtered[col] =
encoder.fit_transform(df_filtered[col])


# Inisialisasi & jalankan Isolation Forest

model = IsolationForest(n_estimators=100,
contamination=0.05, random_state=42)

df_filtered["Anomaly"] =
model.fit_predict(df_filtered)


# Gabungkan kembali dengan data asli

df["Anomaly"] = df_filtered["Anomaly"]
```

```
# Pisahkan anomali & normal

anomali_df = df[df["Anomaly"] == -1]  # Software
yang terdeteksi anomali

normal_df = df[df["Anomaly"] == 1]  # Software
yang dianggap normal


# Simpan hasil

anomali_df.to_csv("Software_Anomalies.csv",
index=False)

print(f"✅ Deteksi anomali selesai! Ditemukan
{len(anomali_df)} software mencurigakan.")
```

4.3.2. One-Class SVM

mengklasifikasi seluruh data sebagai satu kelas dan mencari data yang menyimpang.

```
from sklearn.svm import OneClassSVM

from sklearn.preprocessing import LabelEncoder

import pandas as pd


# Load dataset hasil preprocessing

df =
pd.read_csv("Software_License_Preprocessed.csv")


# Pilih fitur yang akan digunakan
```

```
fitur_terpilih = ["Product_Name", "Publisher",
"License_Code"]

df_filtered = df[fitur_terpilih].copy()


# Encode fitur kategorikal menjadi numerik

encoder = LabelEncoder()

for col in fitur_terpilih:

    df_filtered[col] =
encoder.fit_transform(df_filtered[col])


# Inisialisasi & jalankan One-Class SVM

svm = OneClassSVM(nu=0.05, kernel="rbf",
gamma="scale")  # nu=0.05 berarti 5% data diduga
anomali

df_filtered["Anomaly"] =
svm.fit_predict(df_filtered)


# Gabungkan kembali dengan data asli

df["Anomaly_SVM"] = df_filtered["Anomaly"]


# Pisahkan anomali & normal

anomali_df = df[df["Anomaly_SVM"] == -1]  #
Software yang terdeteksi anomali

normal_df = df[df["Anomaly_SVM"] == 1]  # Software
yang dianggap normal


# Simpan hasil anomali

anomali_df.to_csv("Software_Anomalies_SVM.csv",
index=False)
```

```
print(f"✅ Deteksi anomali selesai! Ditemukan
{len(anomali_df)} software mencurigakan
menggunakan One-Class SVM.")
```

4.3.3. Local Outlier Factor (LOF)

Mengukur densitas lokal tiap data dan membandingkannya dengan tetangganya.

```
from sklearn.neighbors import LocalOutlierFactor

from sklearn.preprocessing import LabelEncoder

import pandas as pd


# Load dataset hasil preprocessing

df =
pd.read_csv("Software_License_Preprocessed.csv")


# Pilih fitur yang akan digunakan

fitur_terpilih = ["Product_Name", "Publisher",
"License_Code"]

df_filtered = df[fitur_terpilih].copy()


# Encode fitur kategorikal menjadi numerik

encoder = LabelEncoder()

for col in fitur_terpilih:

    df_filtered[col] =
encoder.fit_transform(df_filtered[col])
```

```
# Inisialisasi & jalankan Local Outlier Factor
(LOF)

lof = LocalOutlierFactor(n_neighbors=20,
contamination=0.05)

df_filtered["Anomaly"] =
lof.fit_predict(df_filtered)



# Gabungkan kembali dengan data asli

df["Anomaly_LOF"] = df_filtered["Anomaly"]



# Pisahkan anomali & normal

anomali_df = df[df["Anomaly_LOF"] == -1]  #
Software yang terdeteksi anomali

normal_df = df[df["Anomaly_LOF"] == 1]  # Software
yang dianggap normal



# Simpan hasil anomali

anomali_df.to_csv("Software_Anomalies_LOF.csv",
index=False)

print(f"✅ Deteksi anomali selesai! Ditemukan
{len(anomali_df)} software mencurigakan
menggunakan LOF.")
```

4.3.4. Autoencoder

model deep learning yang merekonstruksi input, dan menilai *mean squared error* (MSE) untuk mendeteksi anomali.

```
import pandas as pd

import numpy as np
```

```
import tensorflow as tf

from tensorflow import keras

from sklearn.preprocessing import LabelEncoder,
MinMaxScaler

from sklearn.model_selection import
train_test_split


# Load dataset

df = pd.read_csv("Software_License_Cleaned.csv")


# Pilih fitur yang relevan

fitur_terpilih = ["Product_Name", "Publisher",
"License_Code"]

df_filtered = df[fitur_terpilih].dropna().copy()


# Encode fitur kategorikal menjadi numerik

encoder = LabelEncoder()

for col in fitur_terpilih:

    df_filtered[col] =
encoder.fit_transform(df_filtered[col])


# Normalisasi data ke rentang [0,1]

scaler = MinMaxScaler()

df_scaled = scaler.fit_transform(df_filtered)


# Split dataset (80% training, 20% testing)

X_train, X_test = train_test_split(df_scaled,
test_size=0.2, random_state=42)
```

```
print(f"Dataset training: {X_train.shape},
testing: {X_test.shape}")


# Bangun model Autoencoder

input_dim = X_train.shape[1]


autoencoder = keras.Sequential([

    keras.layers.Dense(32, activation="relu",
input_shape=(input_dim,)),

    keras.layers.Dense(16, activation="relu"),

    keras.layers.Dense(32, activation="relu"),

    keras.layers.Dense(input_dim,
activation="sigmoid")

])


autoencoder.compile(optimizer="adam", loss="mse")


# Train model

history = autoencoder.fit(X_train, X_train,
epochs=50, batch_size=16, validation_data=(X_test,
X_test))


# Bangun model Autoencoder

input_dim = X_train.shape[1]


autoencoder = keras.Sequential([
```

```
    keras.layers.Dense(32, activation="relu",
input_shape=(input_dim,)),

    keras.layers.Dense(16, activation="relu"),

    keras.layers.Dense(32, activation="relu"),

    keras.layers.Dense(input_dim,
activation="sigmoid")

])



autoencoder.compile(optimizer="adam", loss="mse")



# Train model

history = autoencoder.fit(X_train, X_train,
epochs=50, batch_size=16, validation_data=(X_test,
X_test))



# Hitung error rekonstruksi

reconstruction = autoencoder.predict(X_test)

mse = np.mean(np.power(X_test - reconstruction,
2), axis=1)



# Tentukan threshold (gunakan persentil 95 sebagai
cutoff)

threshold = np.percentile(mse, 95)



# Tandai sebagai anomali jika error rekonstruksi
lebih dari threshold

anomalies = mse > threshold



# Simpan hasil anomali
```

```
df_anomalies =
df_filtered.iloc[np.where(anomalies)]

df_anomalies.to_csv("Software_Anomalies_Autoencode
r.csv", index=False)


print(f"✅ Deteksi anomali selesai! Ditemukan
{len(df_anomalies)} software mencurigakan.")
```

4.4. Training dan Prediksi

- Dataset dibagi menjadi data latih dan data uji.
- Model dilatih menggunakan data *unlabeled*.
- Prediksi dilakukan dan hasil klasifikasi (normal atau anomali) diberikan ke setiap entri software.

4.5. Evaluasi & Visualisasi

- Dibuat visualisasi sebaran anomali menggunakan scatter plot, bar chart, dan pie chart.
- Dibandingkan hasil dari keempat model, termasuk perangkat lunak yang terdeteksi oleh ≥ 3 model sebagai kandidat anomali terkuat.

4.6. Penyimpanan Hasil & Dokumentasi

- Hasil deteksi disimpan dalam format `.csv` untuk setiap model.

- Dataset final yang telah didekode (diubah ke nama asli) digunakan untuk pelaporan kepada stakeholder non-teknis.

5. Hasil dan Analisis

Setelah melalui proses preprocessing dan pelatihan model, hasil deteksi anomali diperoleh dari empat metode berbeda, yaitu: **Isolation Forest**, **One-Class SVM**, **Local Outlier Factor (LOF)**, dan **Autoencoder**. Masing-masing metode memberikan output berupa daftar perangkat lunak yang dicurigai sebagai *anomali* berdasarkan pola instalasi dan atribut lisensinya.

5.1. Ringkasan Jumlah Anomali per Model

Model Deteksi Anomali	Jumlah Software Terindikasi Anomali
Isolation Forest	30 entri
One-Class SVM	32 entri
Local Outlier Factor	29 entri
Autoencoder	36 entri

Catatan: Perbedaan jumlah ini mencerminkan sensitivitas dan pendekatan masing-masing model terhadap distribusi data.

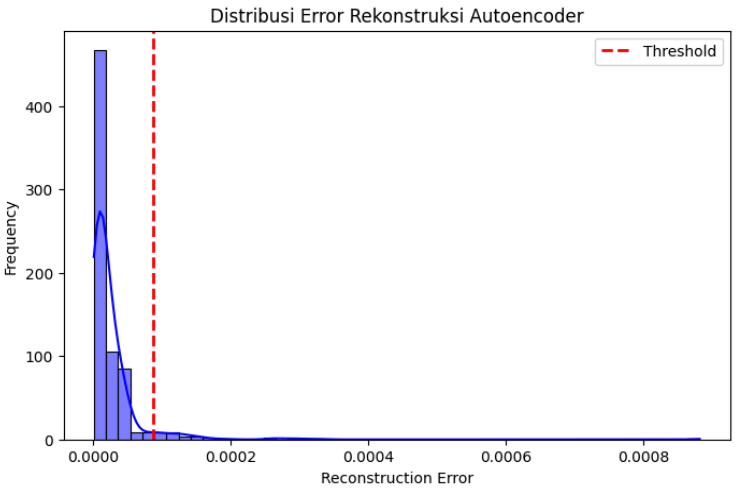
5.2. Karakteristik Umum Software yang Terdeteksi

Dari hasil deteksi keempat model, karakteristik umum software yang diklasifikasikan sebagai anomali antara lain:

Jenis Lisensi Tidak Lengkap atau Tidak Biasa: Banyak entri memiliki License_Code tidak umum atau mengandung placeholder (misalnya: Online License).

- Tipe Software Trial / Freeware: Beberapa software bersifat uji coba atau versi gratis (freeware), yang secara operasional dianggap tidak ideal untuk perangkat resmi perusahaan.
- Perangkat Lunak Default Sistem: Seperti Internet Explorer, yang terinstall otomatis dalam sistem Windows namun tidak relevan untuk dianalisis lebih lanjut.
- Distribusi Tanggal Instalasi Tidak Umum: Misalnya, perangkat lunak yang baru diinstal sangat baru (< 1 bulan) atau sangat lama (> 3 tahun) tanpa update lisensi yang jelas.

Hal ini menunjukkan bahwa model dapat menangkap ketidakwajaran pola penggunaan perangkat lunak dari aspek waktu, tipe, dan legalitas lisensi.



Gambar ini menunjukkan distribusi error rekonstruksi (MSE) dari autoencoder. Cocok untuk menjelaskan bagaimana threshold ditentukan (95th percentile) dan bagaimana error tinggi mengindikasikan anomali.

5.3. Software Terdeteksi oleh ≥ 3 Model

Sebagai langkah lanjutan, dilakukan agregasi dari keempat model untuk mengidentifikasi perangkat lunak yang terdeteksi secara konsisten oleh ≥ 3 model.

Hasilnya menunjukkan sebanyak 29 entri memenuhi kriteria ini dan berikut adalah spesifikasi perangkat lunak yang muncul dalam deteksi gabungan ini:

No.	Product Name	Jumlah Model yang Mendeteksi
1	Adobe Acrobat XI Pro	4
2	CorelDRAW X7	3
3	Microsoft Office Access Runtime (English) 2007	3

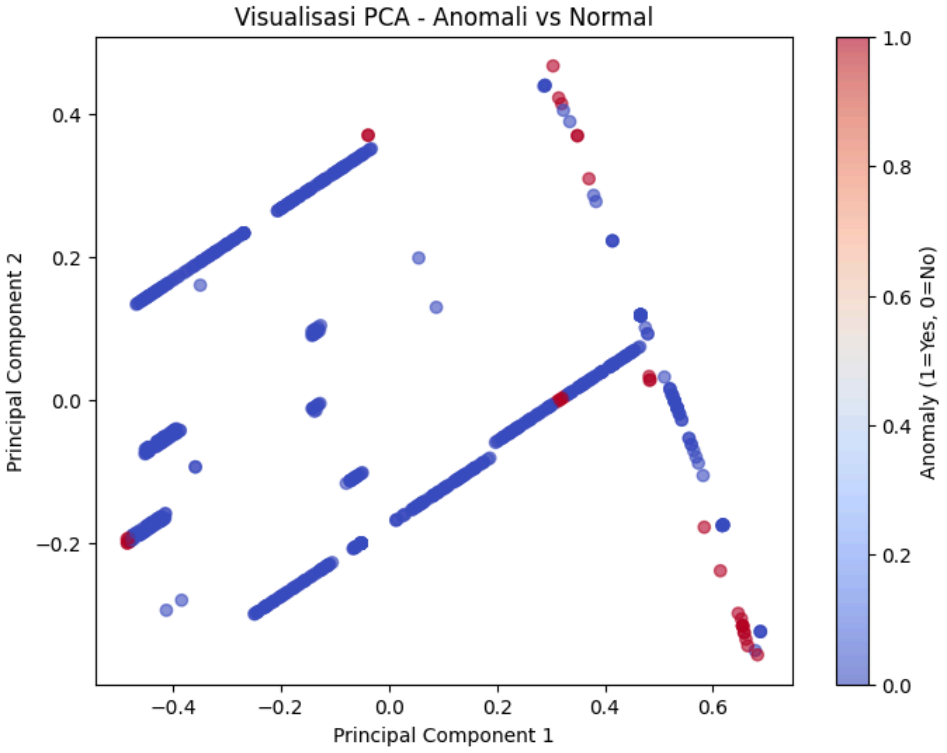
Perangkat lunak di atas terindikasi karena secara konsisten terdeteksi oleh berbagai algoritma dengan pendekatan berbeda.

5.4. Visualisasi Hasil

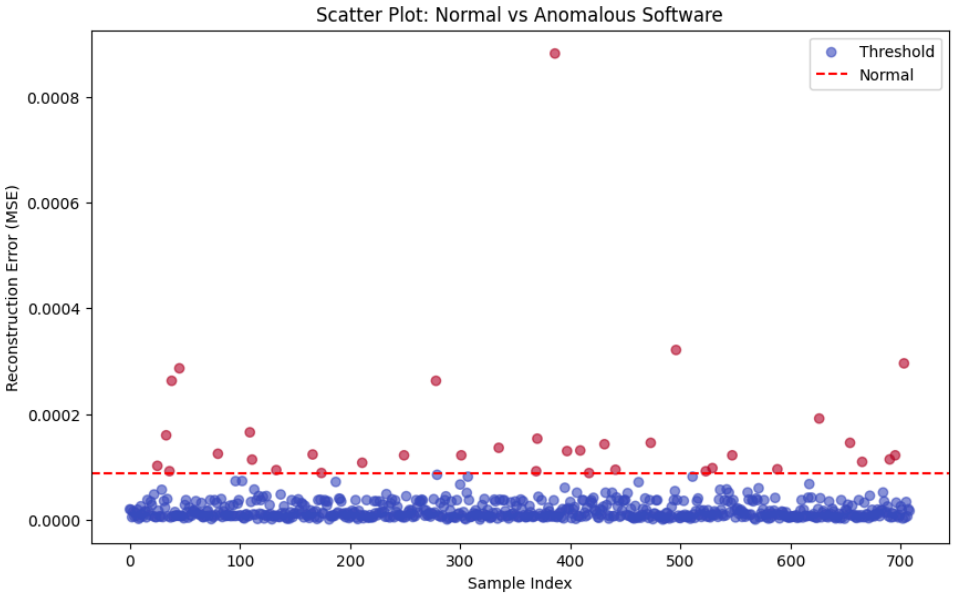
Beberapa bentuk visual yang digunakan:

- Scatter Plot: menunjukkan distribusi normal vs anomali.

- Bar Chart: memperlihatkan software teratas yang paling sering muncul sebagai anomali.



Visualisasi PCA untuk memproyeksikan data berdimensi tinggi (hasil encoding) ke dalam 2D. Membantu menunjukkan distribusi dan clustering antara data normal dan anomali.



Menunjukkan setiap entri dengan error rekonstruksi dan threshold. Cocok digunakan untuk memperkuat penjelasan bahwa nilai di atas threshold = anomali.

6. Kesimpulan dan Rekomendasi

6.1. Kesimpulan

Proyek ini bertujuan untuk mengembangkan sistem deteksi anomali otomatis terhadap data lisensi perangkat lunak di lingkungan perusahaan menggunakan pendekatan machine learning. Dengan menerapkan empat metode berbeda — **Isolation Forest**, **One-Class SVM**, **Local Outlier Factor**, dan **Autoencoder** — diperoleh hasil yang bervariasi namun saling melengkapi.

Beberapa poin penting yang dapat disimpulkan:

- Model *unsupervised learning* terbukti mampu mengenali pola-pola abnormal tanpa pelabelan data manual.

- Tidak semua model menghasilkan daftar anomali yang sama, namun terdapat **29 software** yang secara konsisten ditandai oleh minimal **3 model** sebagai anomali, sehingga layak untuk dianalisis lebih lanjut.

- Autoencoder menunjukkan performa baik dalam mendeteksi pola tersembunyi, terutama pada fitur yang telah dinormalisasi dan dikodekan secara numerik.

- Pendekatan agregasi dari beberapa model membantu meminimalkan false positive dan meningkatkan keakuratan rekomendasi.

6.2. Rekomendasi

Berdasarkan temuan dan pengalaman selama proyek berlangsung, berikut beberapa rekomendasi ke depan:

1. **Validasi Manual**: Lakukan verifikasi terhadap hasil deteksi anomali menggunakan dokumentasi lisensi atau audit software internal perusahaan.

2. **Penerapan Hybrid Model**: Penggabungan hasil dari beberapa model dapat menjadi pendekatan standar, terutama ketika tidak tersedia data label (unsupervised).

3. **Penggunaan Model Supervised**: Jika di masa depan tersedia data software dengan label "legal" atau "ilegal", disarankan untuk melatih model supervised seperti

Random Forest atau XGBoost guna meningkatkan presisi.

4. **Integrasi API Otomatis (Opsional)**: Untuk efisiensi, sistem bisa diintegrasikan dengan API inventaris software seperti Kaseya, guna memeriksa status software secara berkala.

5. **Pembuatan Dashboard Monitoring**: Hasil deteksi dapat divisualisasikan ke dalam bentuk dashboard interaktif untuk tim IT dan audit internal.

7. Daftar Pustaka

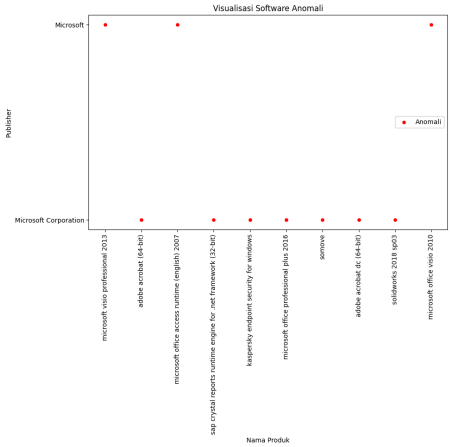
1. Breunig, M. M., Kriegel, H. P., Ng, R. T., & Sander, J. (2000). *LOF: Identifying density-based local outliers*. ACM SIGMOD Record, 29(2), 93–104.

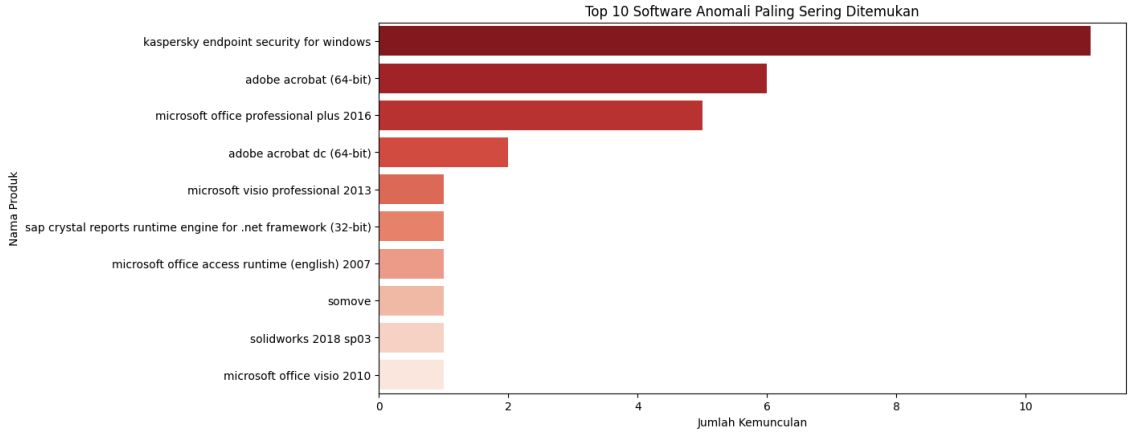
2. Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). *Isolation Forest*. 2008 Eighth IEEE International Conference on Data Mining.

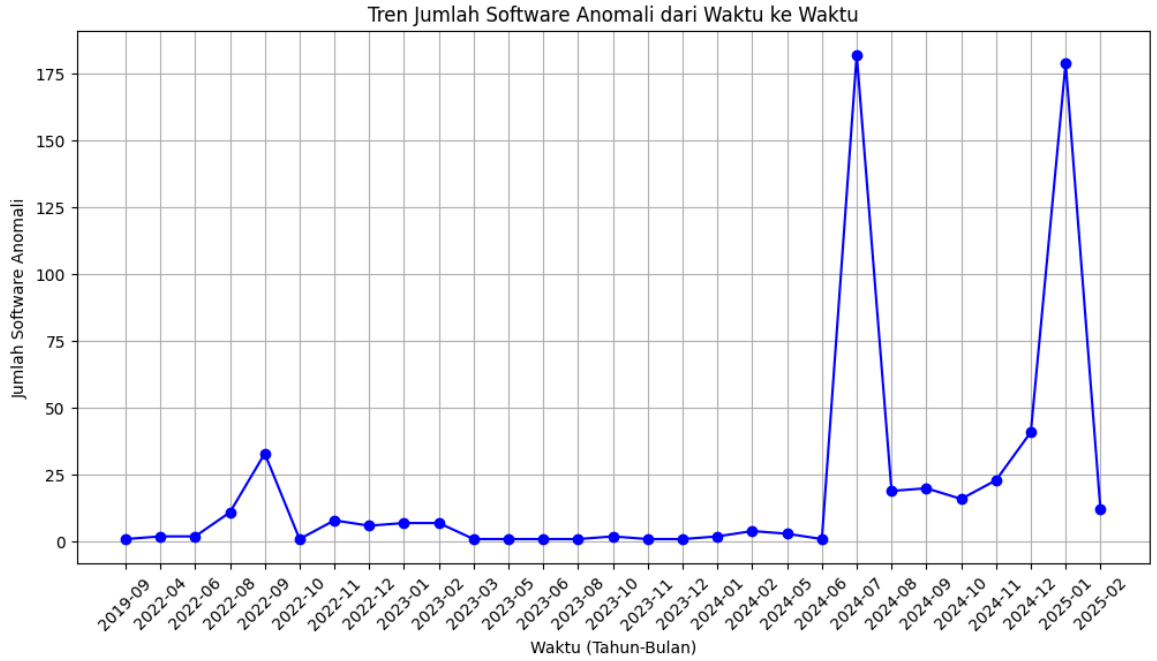
3. Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). *Estimating the support of a high-dimensional distribution*. Neural Computation, 13(7), 1443–1471.

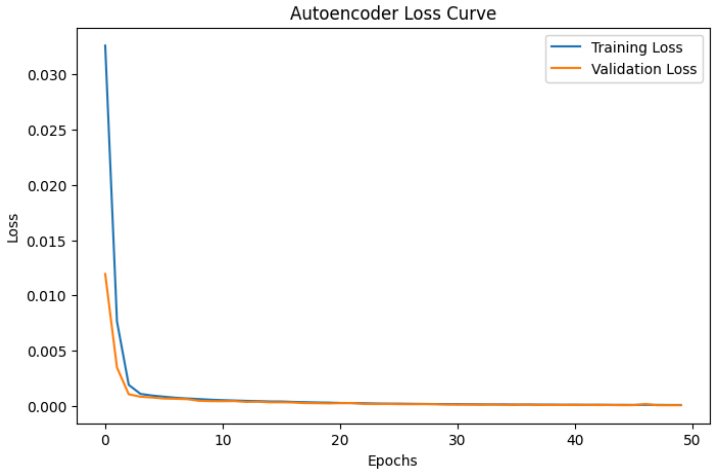
4. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

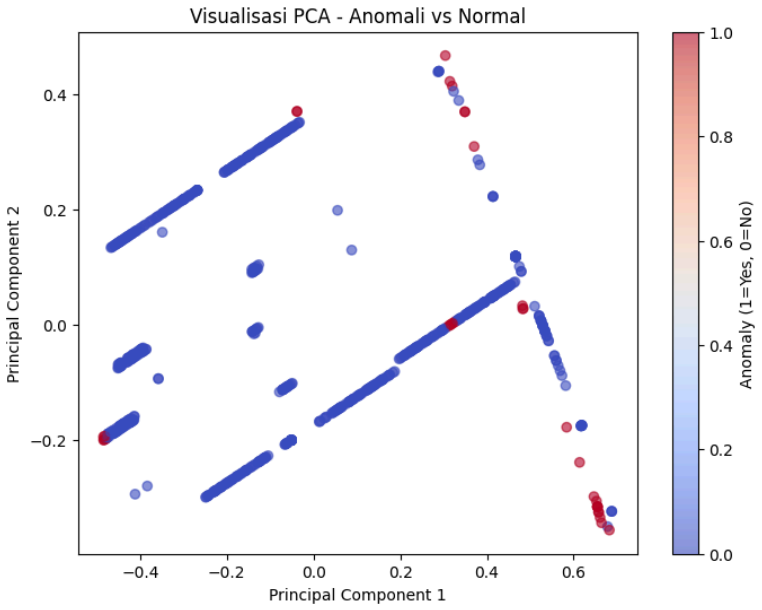
8. Lampiran

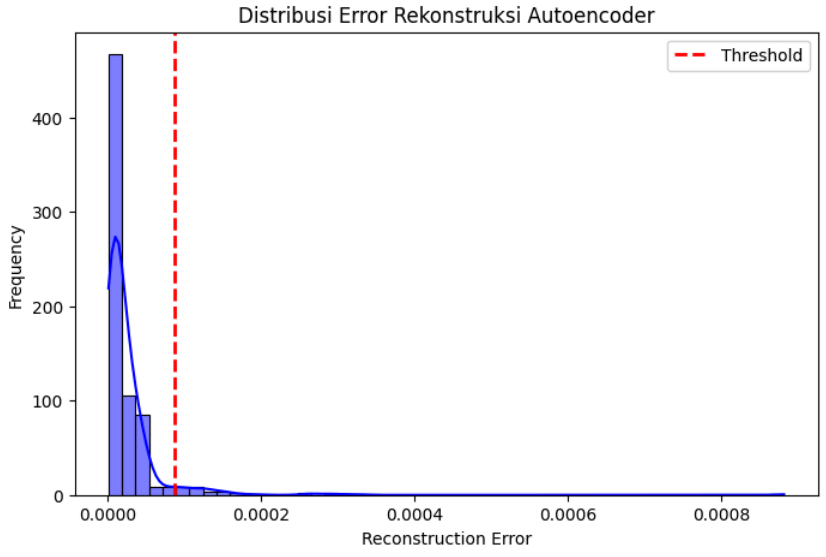

Visualisasi Software Anomali
Microsoft
Microsoft Corporation
Publisher
Anomali
microsoft visio professional 2013
adobe acrobat (64-bit)
microsoft office access runtime (english) 2007
sap crystal reports runtime engine for .net framework (32-bit)
kaspersky endpoint security for windows
microsoft office professional plus 2016
somove
adobe acrobat dc (64-bit)
solidworks 2018 sp03
microsoft office visio 2010
Nama Produk


Top 10 Software Anomali Paling Sering Ditemukan
kaspersky endpoint security for windows
adobe acrobat (64-bit)
microsoft office professional plus 2016
adobe acrobat dc (64-bit)
microsoft visio professional 2013
sap crystal reports runtime engine for .net framework (32-bit)
microsoft office access runtime (english) 2007
somove
solidworks 2018 sp03
microsoft office visio 2010
Nama Produk
0
2
4
6
8
10
Jumlah Kemunculan


Tren Jumlah Software Anomali dari Waktu ke Waktu
Jumlah Software Anomali
0
25
50
75
100
125
150
175
2019-09
2022-04
2022-06
2022-08
2022-09
2022-10
2022-11
2022-12
2023-01
2023-02
2023-03
2023-05
2023-06
2023-08
2023-10
2023-11
2023-12
2024-01
2024-02
2024-05
2024-06
2024-07
2024-08
2024-09
2024-10
2024-11
2024-12
2025-01
2025-02
Waktu (Tahun-Bulan)


Autoencoder Loss Curve
Training Loss
Validation Loss
Loss
Epochs
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
10
20
30
40
50


Visualisasi PCA - Anomali vs Normal
Principal Component 2
Principal Component 1
Anomaly (1=Yes, 0=No)
0.4
0.2
0.0
−0.2
−0.4
−0.2
0.0
0.2
0.4
0.6
1.0
0.8
0.6
0.4
0.2
0.0


Distribusi Error Rekonstruksi Autoencoder
Threshold
Frequency
Reconstruction Error
400
300
200
100
0
0.0000
0.0002
0.0004
0.0006
0.0008