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

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.

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.

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:

1. Sumber Data

- Data yang digunakan berasal dari *Software License Inventory* milik perusahaan, yang mencakup informasi seperti nama software, publisher, license code, tanggal instalasi, dan pengguna terakhir.

- Data tambahan mencakup informasi status software (aktif/tidak aktif), tanggal expired lisensi, dan data yang telah difilter oleh tim internal.

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

2. Proses Preprocessing Data

- Pembersihan data dilakukan untuk menangani *missing values*, menghapus duplikasi, dan menghapus entri software tertentu yang

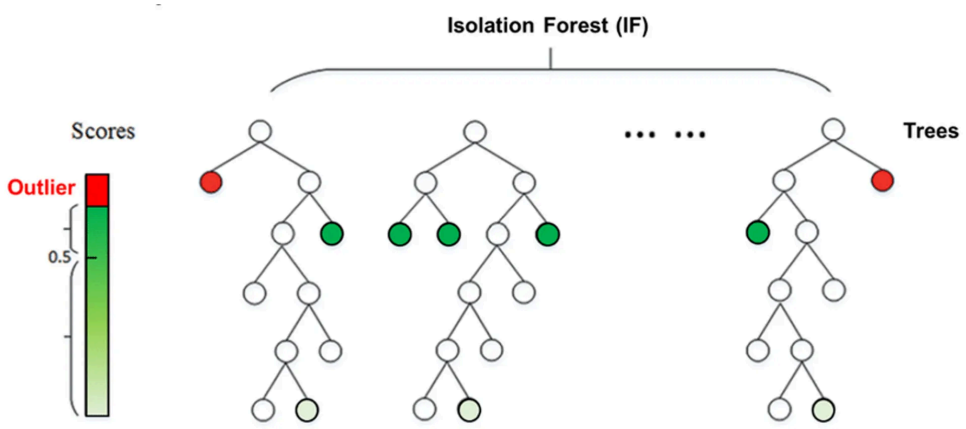
sudah dikonfirmasi bukan anomali (misalnya: Microsoft 365 Online License, Internet Explorer, Cisco AnyConnect).

- Proses encoding dan normalisasi fitur dilakukan agar data dapat digunakan oleh model machine learning.

3. Model yang Digunakan

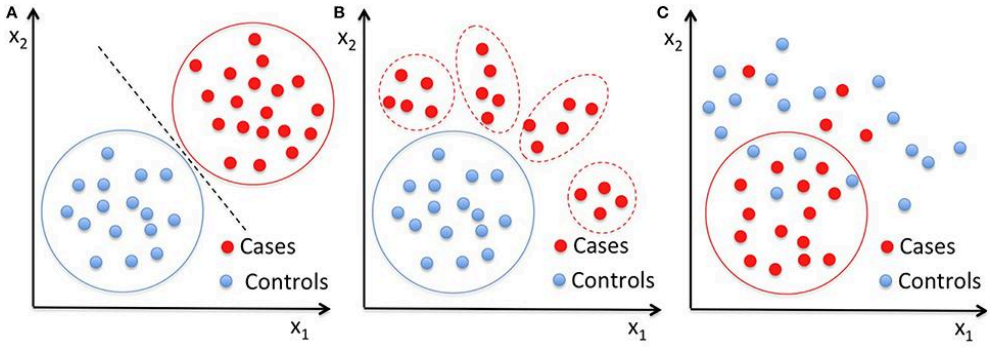
- **Unsupervised Learning Models**:
    - Isolation Forest (IF)

      Mendeteksi anomali dengan cara mengisolasi data secara acak menggunakan pohon keputusan. Data yang mudah diisolasi cenderung anomali.



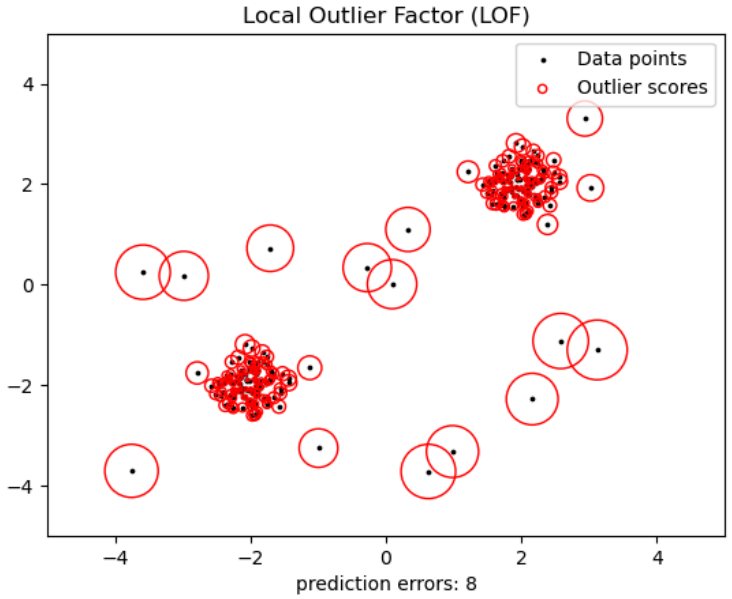
    - One-Class Support Vector Machine (SVM)

      Membangun batas pemisah (*hyperplane*) yang mengelilingi data normal dan mendeteksi anomali sebagai data di luar batas tersebut.



- Local Outlier Factor (LOF)

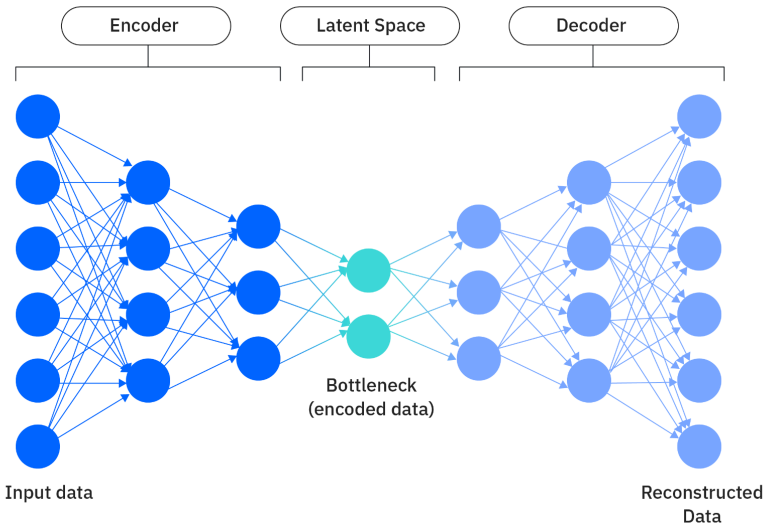
  Mengukur *kepadatan lokal* data. Titik data yang kepadatannya jauh lebih rendah dibandingkan tetangganya dianggap anomali.



- Autoencoder (Deep Learning)

  Neural network yang dilatih untuk merekonstruksi input. Jika model gagal merekonstruksi data tertentu dengan baik (tinggi error), maka data tersebut dianggap anomali. Proyek ini tidak menggunakan metode supervised karena tidak tersedia data

label anomali yang pasti.



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.

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.

Metodologi

Metodologi yang digunakan dalam proyek ini mengikuti alur kerja sistematis untuk mendeteksi software yang berpotensi anomali. Tahapan dilakukan secara berurutan mulai dari persiapan data hingga analisis hasil, dengan pendekatan utama menggunakan *unsupervised learning*.

5.1. Alur Umum Proyek

1. **Pengumpulan Data**
   Data mentah diperoleh dari tim IT perusahaan, berisi informasi inventaris software yang diinstal pada perangkat internal perusahaan.

2. **Preprocessing Data**

   - Pembersihan data: menghapus entri kosong, duplikat, atau tidak relevan.
   - Encoding data kategorik menjadi numerik.
   - Normalisasi data numerik menggunakan Min-Max Scaler.
   - Feature selection: hanya fitur yang dianggap relevan digunakan dalam model (misalnya: `Product_Name`, `Publisher`, `License_Code`, `License_Version`, `Install_Date`).

3. **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	Hasil prediksi digabung, lalu disaring software

	Anomaly	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. **Pemilihan Model Deteksi Anomali**
   Model yang digunakan memiliki pendekatan *unsupervised learning*:

   - **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.")
```

- **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.")
```

- **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.")
```

- **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_Autoencoder.csv"
, index=False)



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

5. **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.

6. **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.
7. **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.

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.

6.1. 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.

6.2. Karakteristik Umum Software yang Terdeteksi

- Banyak dari software anomali memiliki versi lisensi yang tidak lengkap, tidak umum, atau sangat baru/lama dari segi `Install Date`.
- Beberapa perangkat lunak yang muncul berulang kali dalam hasil model adalah software yang bersifat trial, *freeware*, atau menggunakan model lisensi online.
- Muncul di berbagai model termasuk:
    - *Adobe Acrobat XI Pro*
    - *CorelDRAW*
    - *Microsoft Visio Professional*

6.3. Software Terdeteksi oleh ≥ 3 Model

Untuk meningkatkan keyakinan atas hasil, dilakukan agregasi terhadap semua hasil model. Software yang muncul pada minimal tiga dari empat model dianggap sebagai anomali yang sangat mencurigakan.

- Total perangkat lunak terdeteksi oleh ≥ 3 model: 3 entri unik
- Visualisasi seperti pie chart dan stacked bar digunakan untuk membandingkan distribusi hasil tiap model.

6.4. 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.

Kesimpulan dan Rekomendasi

7.1. 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.

7.2. 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.

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.

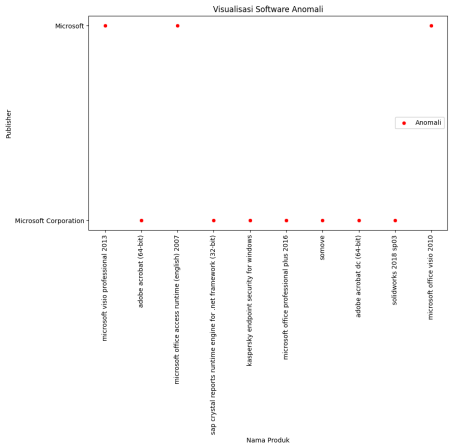
2. Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008). *Isolation Forest*. 2008 Eighth IEEE International Conference on Data Mining.

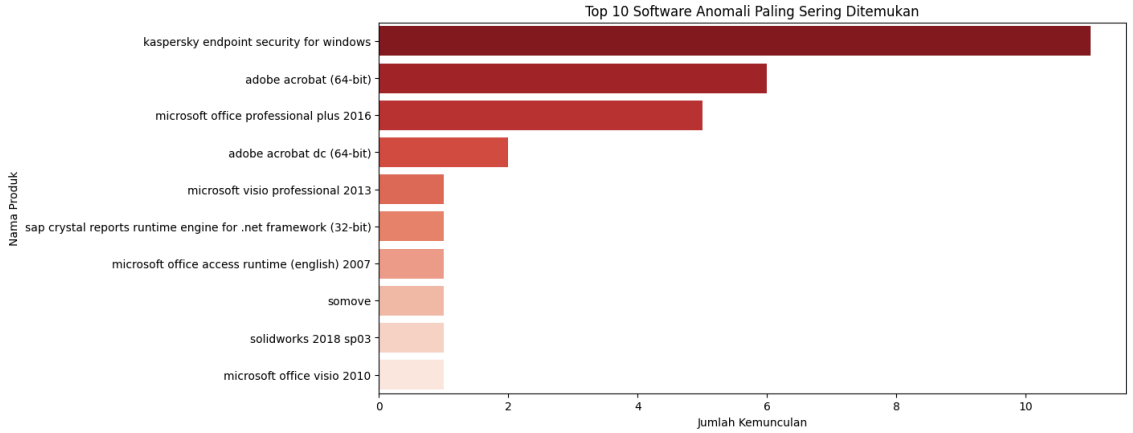
3. Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). *Estimating the support of a high-dimensional distribution*. Neural Computation, 13(7), 1443–1471.

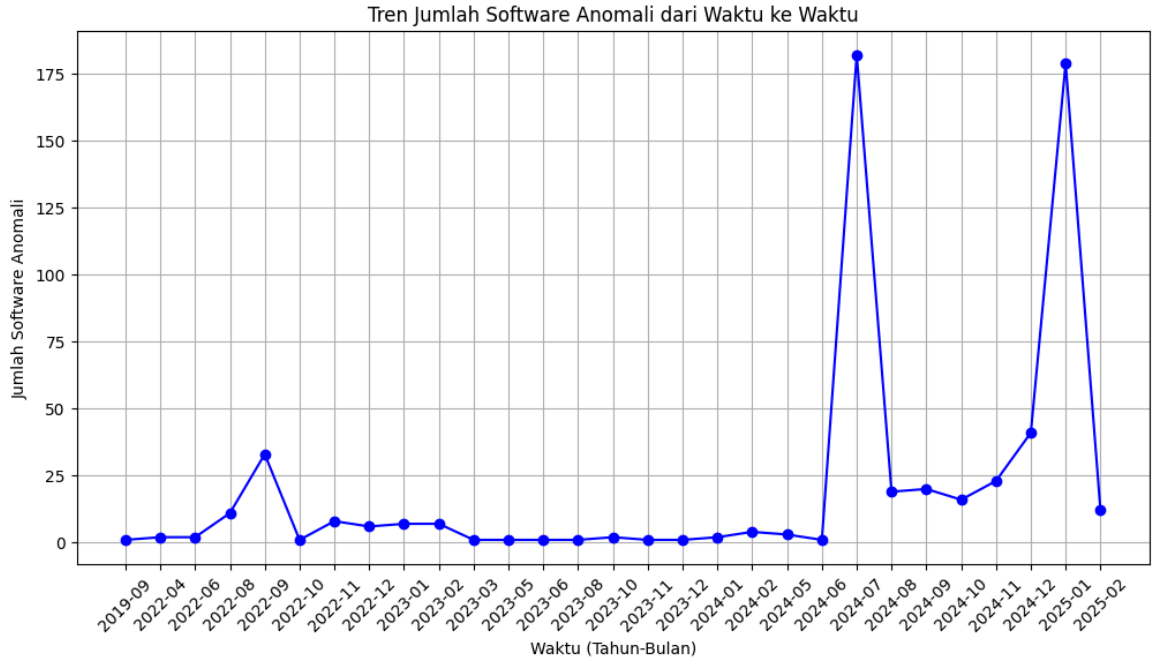
4. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

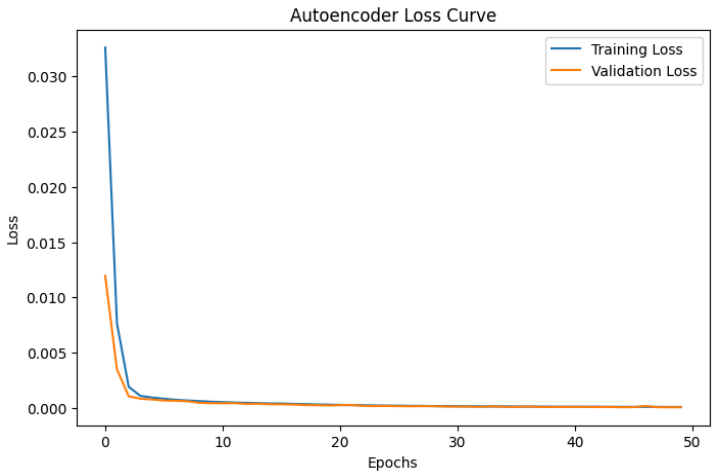
5. Dokumentasi Scikit-Learn: https://scikit-learn.org/stable/

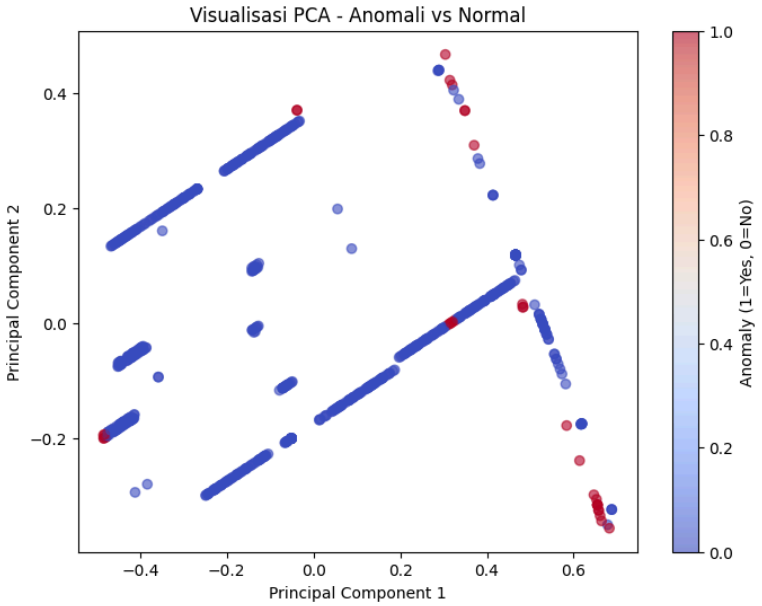
Lampiran

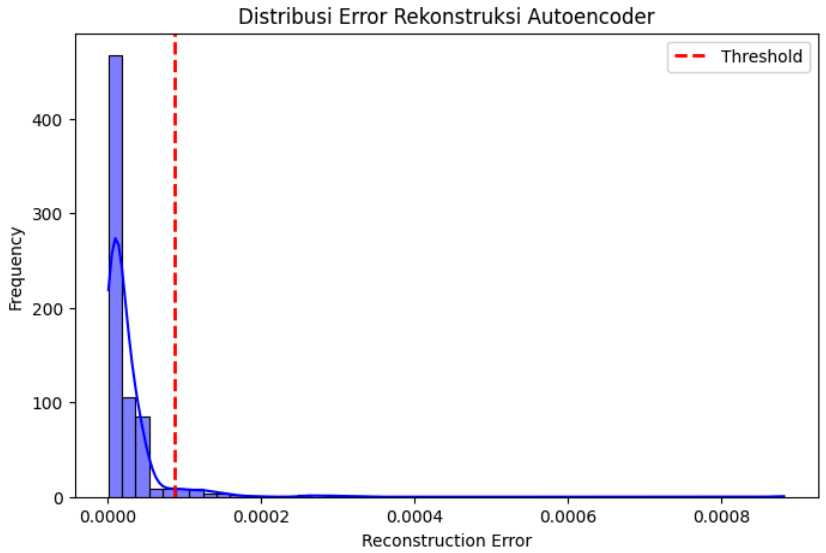

Visualisasi Software Anomali
Publisher
Microsoft
Microsoft Corporation
Anomali
microsoft visio professional 2013
adobe acrobat (64-bit)
microsoft office access runtime (english) 2007
sap crystal reports runtime engine for .net framework (32-bit)
kaspersky endpoint security for windows
microsoft office 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