TUGAS MATA KULIAH

SISTEM CERDAS

(Model Deteksi Anomali Log Sistem)

Disusun Oleh :

Kevin Sipahutar	2210631250017
Fahry Firdaus Marpaung	2210631250048
Rama Diaz	2210631250067

PROGAM STUDI SISTEM INFORMASI

FAKULTAS ILMU KOMPUTER

UNIVERSITAS SINGAPERBANGSA KARAWANG

2025

A. Pendahuluan

Secara umum, terdapat pendekatan dalam Machine Learning utama sebagai berikut:

a. Supervised Learning

Supervised learning adalah jenis pembelajaran mesin yang dilatih dengan data berlabel, yang berarti setiap input memiliki output yang sesuai (label) yang diketahui.

b. Unsupervised Learning

Supervised learning adalah jenis pembelajaran mesin yang dilatih dengan data tidak berlabel, di mana model belajar dari pola data tanpa instruksi atau supervisi khusus.

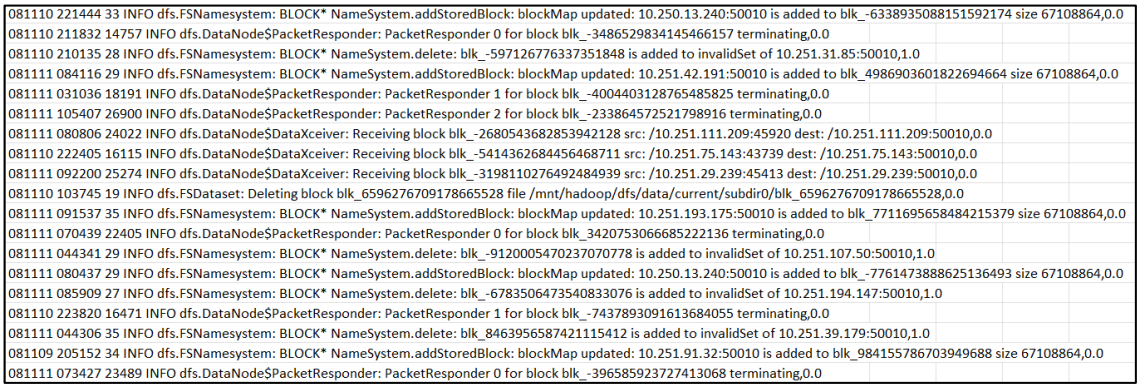
Pada tugas Sistem Cerdas ini kelompok kami menggunakan pendekatan Supervised Learning untuk membuat model. Kelompok kami membuat model untuk mendeteksi sebuah anomali yang terdapat pada suatu sistem log yang mana ini termasuk ke dalam jenis Deteksi Anomali. Deteksi anomali (anomaly detection) adalah salah satu jenis tugas dalam machine learning yang bertujuan untuk mengidentifikasi data atau pola yang menyimpang secara signifikan dari mayoritas data lainnya. Data yang menyimpang ini disebut sebagai anomali, outlier, atau abnormal. Lalu kami menggunakan algoritma Isolation Forest, algoritma ini digunakan untuk menyelesaikan tujuan dari deteksi anomali. Algoritma ini bekerja dengan cara mengisolasi setiap titik data secara acak dalam struktur pohon, dan mengukur seberapa cepat suatu titik dapat diisolasi. Semakin cepat suatu data terisolasi, semakin besar kemungkinan data tersebut merupakan anomali.

B. Dataset

Kami menggunakan sebuah dataset yang tersedia secara publik. Kami menggunakan dataset dari kaggle yang dapat dibuka pada link berikut ini.

https://www.kaggle.com/datasets/krishd123/log-data-for-anomaly-detection

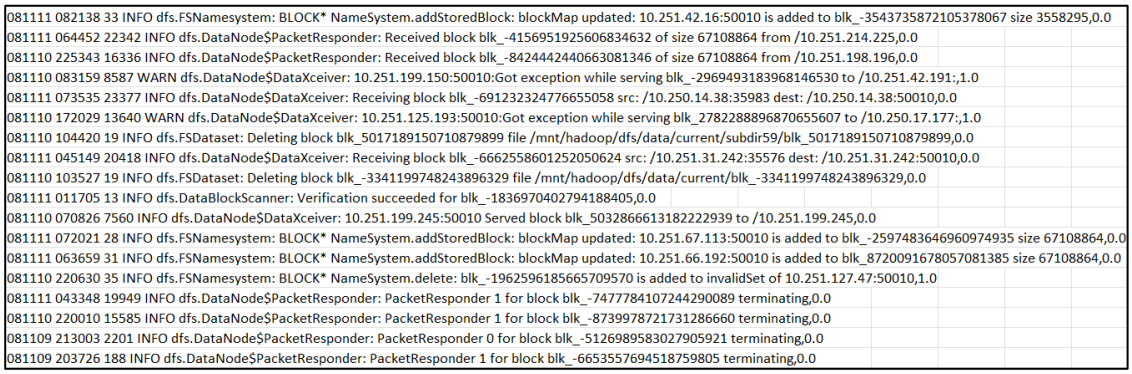
- Data Training



```
081110 221444 33 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.250.13.240:50010 is added to blk_-6338935088151592174 size 67108864,0.0
081110 211832 14757 INFO dfs.DataNode$PacketResponder: PacketResponder 0 for block blk_-3486529834145466157 terminating,0.0
081110 210135 28 INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_-597126776337351848 is added to invalidSet of 10.251.31.85:50010,1.0
081111 084116 29 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.42.191:50010 is added to blk_4986903601822694664 size 67108864,0.0
081111 031036 18191 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_-4004403128765485825 terminating,0.0
081111 105407 26900 INFO dfs.DataNode$PacketResponder: PacketResponder 2 for block blk_-233864572521798916 terminating,0.0
081111 080806 24022 INFO dfs.DataNode$DataXceiver: Receiving block blk_-2680543682853942128 src: /10.251.111.209:45920 dest: /10.251.111.209:50010,0.0
081110 222405 16115 INFO dfs.DataNode$DataXceiver: Receiving block blk_-5414362684456468711 src: /10.251.75.143:43739 dest: /10.251.75.143:50010,0.0
081111 092200 25274 INFO dfs.DataNode$DataXceiver: Receiving block blk_-3198110276492484939 src: /10.251.29.239:45413 dest: /10.251.29.239:50010,0.0
081110 103745 19 INFO dfs.FSDataset: Deleting block blk_6596276709178665528 file /mnt/hadoop/dfs/data/current/subdir0/blk_6596276709178665528,0.0
081111 091537 35 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.193.175:50010 is added to blk_7711695658484215379 size 67108864,0.0
081111 070439 22405 INFO dfs.DataNode$PacketResponder: PacketResponder 0 for block blk_3420753066685222136 terminating,0.0
081111 044341 29 INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_-9120005470237070778 is added to invalidSet of 10.251.107.50:50010,1.0
081111 080437 29 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.250.13.240:50010 is added to blk_-7761473888625136493 size 67108864,0.0
081111 085909 27 INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_-6783506473540833076 is added to invalidSet of 10.251.194.147:50010,1.0
081110 223820 16471 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_-7437893091613684055 terminating,0.0
081111 044306 35 INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_8463956587421115412 is added to invalidSet of 10.251.39.179:50010,1.0
081109 205152 34 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.91.32:50010 is added to blk_984155786703949688 size 67108864,0.0
081111 073427 23489 INFO dfs.DataNode$PacketResponder: PacketResponder 0 for block blk_-396585923727413068 terminating,0.0
```

Kami akan melakukan pelatihan data menggukan dataset tersebut. Dataset ini berjumlah sebanyak 1.048.576 data.

- Data Testing



```
081111 082138 33 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.42.16:50010 is added to blk_-3543735872105378067 size 3558295,0.0
081111 064452 22342 INFO dfs.DataNode$PacketResponder: Received block blk_-4156951925606834632 of size 67108864 from /10.251.214.225,0.0
081110 225343 16336 INFO dfs.DataNode$PacketResponder: Received block blk_-8424442440663081346 of size 67108864 from /10.251.198.196,0.0
081110 083159 8587 WARN dfs.DataNode$DataXceiver: 10.251.199.150:50010:Got exception while serving blk_-2969493183968146530 to /10.251.42.191:,1.0
081111 073535 23377 INFO dfs.DataNode$DataXceiver: Receiving block blk_-691232324776655058 src: /10.250.14.38:35983 dest: /10.250.14.38:50010,0.0
081110 172029 13640 WARN dfs.DataNode$DataXceiver: 10.251.125.193:50010:Got exception while serving blk_2782288896870655607 to /10.250.17.177:,1.0
081110 104420 19 INFO dfs.FSDataset: Deleting block blk_5017189150710879899 file /mnt/hadoop/dfs/data/current/subdir59/blk_5017189150710879899,0.0
081111 045149 20418 INFO dfs.DataNode$DataXceiver: Receiving block blk_-6662558601252050624 src: /10.251.31.242:35576 dest: /10.251.31.242:50010,0.0
081110 103527 19 INFO dfs.FSDataset: Deleting block blk_-3341199748243896329 file /mnt/hadoop/dfs/data/current/blk_-3341199748243896329,0.0
081111 011705 13 INFO dfs.DataBlockScanner: Verification succeeded for blk_-1836970402794188405,0.0
081110 070826 7560 INFO dfs.DataNode$DataXceiver: 10.251.199.245:50010 Served block blk_5032866613182222939 to /10.251.199.245,0.0
081111 072021 28 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.67.113:50010 is added to blk_-2597483646960974935 size 67108864,0.0
081111 063659 31 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.66.192:50010 is added to blk_8720091678057081385 size 67108864,0.0
081110 220630 35 INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_-1962596185665709570 is added to invalidSet of 10.251.127.47:50010,1.0
081111 043348 19949 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_-7477784107244290089 terminating,0.0
081110 220010 15585 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_-8739978721731286660 terminating,0.0
081109 213003 2201 INFO dfs.DataNode$PacketResponder: PacketResponder 0 for block blk_-5126989583027905921 terminating,0.0
081109 203726 188 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_-6653557694518759805 terminating,0.0
```

Kami akan melakukan pengujian menggunakan data tersebut. Data ini juga berjumlah sebanyak 1.048.576 data.

C. Code Program

- Download Dataset dan Mempersiapkan Data log HDFS (Hadoop Distributed File System)
  - Import Libraries & Dependencies

```
import os
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import kaggle
import zipfile
import io
```

- Membuat Direktori Data

```
def download_hdfs_dataset():

    os.makedirs('data', exist_ok=True)
```

"os.makedirs('data', exist_ok=True)" digunakan untuk membuat folder bernama 'data' di direktori saat ini, lalu "exist_ok=True", parameter ini mencegah error jika folder sudah ada.

- Download Dataset dari Kaggle

```
print("Mengunduh dataset dari Kaggle...")
        try:
            kaggle.api.dataset_download_files(
                'krishd123/log-data-for-anomaly-detection',
                path='data',
                unzip=True
            )
            print("Dataset berhasil diunduh dan diekstrak!")
        except Exception as e:
            print(f"Error saat mengunduh dataset: {str(e)}")
            print("\nPastikan Anda telah:")
            print("1. Menginstal paket kaggle: pip install kaggle")
            print("2. Menempatkan kaggle.json di direktori ~/.config/kaggle/")
            print("3. Mengatur izin yang benar: chmod 600 ~/.config/kaggle/kaggle.json")
            return False
        if not os.path.exists(target_file_path):
            print(f"File {target_file_path} tidak ditemukan setelah ekstraksi.")
            return False
    print(f"File {target_file_path} berhasil ditemukan!")
    return True
```

Code di atas bertujuan untuk melakukan installasi dataset dari laman kaggle. Setelah itu terdapat Error Handling yang mana akan menangkap semua exception yang mungkin terjadi saat proses downloading. Setelah itu terdapat Validasi setalah proses download di akhir.

- Membaca File Log

```
def prepare_dataset():

    log_file_path = os.path.join('data', 'hdfs_log', 'hdfs.log', 'sorted.log')
    with open(log_file_path, 'r', encoding='utf-8', errors='ignore') as f:
        logs = f.readlines()
    df = pd.DataFrame({'log': [log.strip() for log in logs]})
    print(f"Total entri log: {len(df)}")
```

Code di atas berfungsi untuk membaca log dengan detail sebagai berikut:

1) with open(...) as f:: Context manager untuk membuka file dengan aman
2) encoding='utf-8': Encoding untuk membaca karakter Unicode
3) errors='ignore': Mengabaikan karakter yang tidak bisa di-decode
4) f.readlines(): Membaca semua baris dalam file sebagai list

5) log.strip(): Menghilangkan whitespace (spasi, newline) di awal dan akhir setiap baris
6) pd.DataFrame({'log': [...]}): Membuat DataFrame dengan kolom 'log'

- Mendefinisikan Pola Anomali

```
anomaly_patterns = [
        'Exception',
        'Error',
        'Failed',
        'Timeout',
        'Connection refused',
        'Permission denied',
        'OutOfMemory',
        'NullPointerException',
        'StackOverflowError',
        'ClassNotFoundException',
        'WARNING',
        'ERROR',
        'CRITICAL',
        'Fatal',
        'Invalid',
        'Missing',
        'Not found',
        'Access denied',
        'Authentication failed',
        'Connection lost'
    ]
```

Code di atas adalah List berisi 20 pattern string yang mengindikasikan masalah dalam sistem.

- Labeling Data

```
labels = np.zeros(len(df))
   for i, log in enumerate(df['log']):
       if any(pattern.lower() in str(log).lower() for pattern in anomaly_patterns):
           labels[i] = 1
```

Code di atas digunakan untuk melakukan labeling pada dataset. Berikut adalah penjelasannya:

1) np.zeros(len(df)): Membuat array berisi 0 sebanyak jumlah log entries
2) enumerate(df['log']): Iterasi dengan index dan value setiap log entry
3) str(log).lower(): Convert log ke string dan lowercase untuk case-insensitive matching
4) pattern.lower(): Convert pattern ke lowercase
5) any(...): Return True jika minimal satu pattern ditemukan dalam log
6) labels[i] = 1: Set label menjadi 1 (anomali) jika pattern ditemukan

- Menambahkan Label ke data frame dan melakukan split dataset

```
df['label'] = labels

train_df, test_df = train_test_split(df, test_size=0.2, random_state=42)
```

”df[‘label’] = labels” berfungsi untuk Menambahkan kolom 'label' ke DataFrame yang berisi array labels. “train_df, test_df” untuk melakukan pembagian dataset untuk dibagi ke data train dan data testing.

- Menyimpan Dataset

```
train_df.to_csv('data/train_logs.csv', index=False)
test_df.to_csv('data/test_logs.csv', index=False)
```

Selanjut code ini berfungsi untuk menyimpan data train ke dalam direktori data dengan nama train_log.csv dan menyimpan data testing ke dalam direktori data dengan nama test_log.csv.

➢ Tahap Preprocessing

- Import Libraries

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
import re
from tqdm import tqdm
from datetime import datetime
```

- Class Definiton dan Constructor

```
class LogPreprocessor:
    def __init__(self):
        self.scaler = StandardScaler()
```

1) Class LogPreprocessor: Encapsulates semua functionality untuk preprocessing log
2) self.scaler: Instance dari StandardScaler yang akan digunakan untuk normalisasi
3) StandardScaler: Mengubah data menjadi distribusi normal dengan mean=0 dan std=1
4) Keuntungan OOP: Scaler dapat di-reuse dan disimpan untuk konsistensi preprocessing

- Method extract_features()

```
def extract_features(self, log_line):

    features = {}
```

Terdapat beberapa fungsi pada method ini, berikut penjelasannya:

1. Ekstraksi Log Level

```
log_levels = ['INFO', 'WARNING', 'ERROR', 'CRITICAL', 'DEBUG', 'FATAL']
    features['log_level'] = 'UNKNOWN'
    for level in log_levels:
        if level in log_line:
            features['log_level'] = level
            break
```

   1) log_levels: Array berisi 6 level log standar
   2) Default value: 'UNKNOWN' jika tidak ada level yang ditemukan
   3) Prioritas: Level pertama yang ditemukan akan digunakan (INFO > WARNING > ERROR, dst.)

2. Ekstraksi Pola Error

```
error_patterns = [
        'Exception', 'Error', 'Failed', 'Timeout', 'Connection refused',
        'Permission denied', 'OutOfMemory', 'NullPointerException',
        'StackOverflowError', 'ClassNotFoundException', 'WARNING',
        'ERROR', 'CRITICAL', 'Fatal', 'Invalid', 'Missing',
        'Not found', 'Access denied', 'Authentication failed',
        'Connection lost'
    ]

    features['error_count'] = sum(1 for pattern in error_patterns if pattern.lower() in log_line.lower())
```

   1) error_patterns: 20 keyword yang mengindikasikan masalah sistem
   2) Case-insensitive matching: pattern.lower() in log_line.lower()
   3) Counting approach: Menghitung berapa banyak error pattern dalam satu log line
   4) Generator expression: sum(1 for pattern in ...) - efficient memory usage
   5) Output: Integer (0 = tidak ada error pattern, >0 = ada indikasi masalah)

3. Ekstraksi Panjang Pesan ,Karakter Khusus, Angka, dan Block ID

```
features['message_length'] = len(log_line)

features['special_chars'] = len(re.findall(r'[^a-zA-Z0-9\s]', log_line))

features['numbers'] = len(re.findall(r'\d+', log_line))

block_pattern = r'blk_[-]?\d+'
    features['block_count'] = len(re.findall(block_pattern, log_line))
```

- Method preprocess_logs()

```
def preprocess_logs(self, log_data, progress_bar=False):
```

Terdapat beberapa fungsi pada method ini, berikut penjelasannya:

1. Feature Extraction dengan Progress Bar

```
features_list = [self.extract_features(log) for log in tqdm(log_data, desc="Memproses log")]
    else:
        features_list = [self.extract_features(log) for log in log_data]
    df = pd.DataFrame(features_list)
```

   1) List comprehension: Efficient way untuk apply function ke setiap element
   2) tqdm: Progress bar library dengan description "Memproses log"
   3) Conditional execution: Progress bar hanya jika diminta (untuk performance)
   4) DataFrame creation: Convert list of dictionaries menjadi structured DataFrame

2. Categorical Variable Encoding

```
if 'log_level' in df.columns:
        df = pd.get_dummies(df, columns=['log_level'])
```

Code di atas akan mengkonversi variabel kategorikal

3. Feature Scalling/Normalization

```
numerical_cols = df.select_dtypes(include=[np.number]).columns
df[numerical_cols] = self.scaler.fit_transform(df[numerical_cols])
```

   1) select_dtypes(): Memilih hanya kolom dengan tipe data numerik
   2) StandardScaler: Normalisasi dengan formula: (x - mean) / std
   3) Benefit: Semua fitur numerik memiliki skala yang sama (mean=0, std=1)
   4) ML importance: Mencegah fitur dengan range besar mendominasi model

- Utility Methods

1. Save Scaler

```
def save_scaler(self, path):

    import joblib
    joblib.dump(self.scaler, path)
```

   1) joblib: Library untuk serialization yang efficient untuk scikit-learn objects

2) Purpose: Menyimpan fitted scaler untuk konsistensi preprocessing
3) Use case: Training phase simpan scaler, inference phase load scaler yang sama

2. Load Scaler

```
def load_scaler(self, path):

        import joblib
        self.scaler = joblib.load(path)
```

1) Load pre-fitted scaler: Untuk consistency antara training dan inference
2) Critical: Harus menggunakan scaler yang sama yang digunakan saat training
3) Data leakage prevention: Tidak fit ulang scaler pada test data

➢ Pembuatan Model Pendeteksi Anomali

- Import Libraries

```
import numpy as np
from sklearn.ensemble import IsolationForest
import joblib
import os
```

- Class AnomalyDetector

```
class AnomalyDetector:
```

Terdapat beberapa fungsi, dapat dilihat sebagai berikut:

1. Constructor

```
def __init__(self, contamination=0.5, threshold_percentile=50):
```

1) Inisialisasi model IsolationForest dengan 100 estimator dan 4 core
2) contamination: proporsi data yang dianggap anomali (default 50%)
3) threshold_percentile: persentil untuk menentukan ambang batas
4) threshold = None: akan diisi saat training

2. Training Method

```
def train(self, X):
        self.model.fit(X)

        scores = -self.model.score_samples(X)

        mean_score = np.mean(scores)
        std_score = np.std(scores)
        percentile_threshold = np.percentile(scores, self.threshold_percentile)

        self.threshold = min(percentile_threshold, mean_score + std_score)

        print(f"Ambang batas anomali ditetapkan ke: {self.threshold:.4f}")
        print(f"Jumlah anomali dalam set pelatihan: {np.sum(scores > self.threshold)}")
```

1) Model Fitting: self.model.fit(X) melatih IsolationForest dengan data X
2) Score Calculation: score_samples(X) menghasilkan anomaly score (nilai negatif = lebih anomali)
3) Score Inversion: Menggunakan - untuk membuat nilai positif = lebih anomali
4) Statistik: mean + std (deteksi outlier statistik)
5) Persentil: Persentil ke-50 dari distribusi score
6) Final Threshold: Mengambil nilai minimum dari keduanya (lebih konservatif)

3. Predict Method

```
def predict(self, X):
    scores = -self.model.score_samples(X)
    return (scores > self.threshold).astype(int)
```

1) Menghitung anomaly score untuk data baru
2) Membandingkan dengan threshold
3) Mengembalikan array binary: 1 = anomali, 0 = normal

4. Predict Probability Method

```
def predict_proba(self, X):
    return -self.model.score_samples(X)
```

1) Mengembalikan continuous anomaly score
2) Semakin tinggi nilai = semakin anomali
3) Berguna untuk ranking dan analisis lebih detail

5. Save Model Method

```
def save_model(self, path):
    os.makedirs(os.path.dirname(path), exist_ok=True)
    model_data = {
        'model': self.model,
        'threshold': self.threshold,
        'threshold_percentile': self.threshold_percentile
    }
    joblib.dump(model_data, path)
```

1) Membuat direktori jika belum ada
2) Mengemas data dalam dictionary:
    a. Model IsolationForest yang sudah dilatih
    b. Threshold yang telah dihitung
    c. Parameter threshold_percentile
3) Menyimpan menggunakan joblib (format pickle yang efisien)

6. Load Model Method

```
def load_model(self, path):
    model_data = joblib.load(path)
    self.model = model_data['model']
    self.threshold = model_data['threshold']
    self.threshold_percentile = model_data['threshold_percentile']
```

1) Memuat dictionary dari file
2) Restore semua komponen ke instance saat ini
3) Model siap digunakan untuk prediksi

➢ Melakukan Training Model

- Import Libraries

```
import pandas as pd
from preprocessing import LogPreprocessor
from model import AnomalyDetector
import os
import logging
import time
from tqdm import tqdm
```

- Logging Configuration

```
def train_model():

    logging.basicConfig(
        level=logging.INFO,
        format='%(asctime)s [%(levelname)s] %(message)s',
        datefmt='%Y-%m-%d %H:%M:%S'
    )
```

1) level=logging.INFO: Set minimum level ke INFO (INFO, WARNING, ERROR, CRITICAL)
2) format: Template pesan log dengan timestamp, level, dan pesan
3) Contoh output: 2024-01-15 14:30:25 [INFO] Memuat dataset pelatihan...
4) datefmt: Format timestamp (YYYY-MM-DD HH:MM:SS)

- Performance Monitoring Setup

```
start_time = time.time()
    logging.info("Memuat dataset pelatihan...")
```

1) start_time: Catat waktu mulai untuk menghitung total training time
2) time.time(): Unix timestamp dalam detik (floating point)
3) Logging info: Memberikan feedback ke user bahwa proses dimulai

- Dataset Loading

```
test_df = pd.read_csv('data/train_logs.csv')
logging.info(f"Memuat {len(test_df)} entri log.")
```

1) Variable naming issue: test_df seharusnya train_df (kemungkinan typo)
2) File path: data/train_logs.csv - file yang dibuat oleh download_dataset.py
3) DataFrame structure: Berisi kolom 'log' (text) dan 'label' (0/1)
4) Logging: Memberikan informasi jumlah data yang dimuat

- Anomaly Ratio Calculation

```
anomaly_ratio = test_df['label'].mean()
logging.info(f"Rasio anomali dalam dataset: {anomaly_ratio:.4f}")
```

1) anomaly_ratio: Persentase label=1 dalam dataset
2) test_df['label'].mean(): Rata-rata binary labels (0 dan 1)
3) Contoh: Jika 100 data, 15 anomali → ratio = 0.1500
4) Format output: 4 decimal places untuk precision
5) Kegunaan: Menentukan parameter contamination untuk model

- Component Initialization

```
preprocessor = LogPreprocessor()
detector = AnomalyDetector(
    contamination=max(anomaly_ratio, 0.1),  # Gunakan rasio aktual atau minimum 0.1
    threshold_percentile=50
   )
```

1) AnomalyDetector Parameters:

   contamination:

   a. max(anomaly_ratio, 0.1): Minimum 10% untuk stability
   b. Jika dataset punya 5% anomali, tetap gunakan 10%

c. Jika dataset punya 20% anomali, gunakan 20%

- Data Training Process

```
logging.info("Memproses data pelatihan (ekstraksi fitur dan penskalaan)...")
    t0 = time.time()
    processed_data = preprocessor.preprocess_logs(test_df['log'].tolist(), progress_bar=True)
    logging.info(f"Bentuk data yang diproses: {processed_data.shape}")
    logging.info(f"Pemrosesan selesai dalam {time.time() - t0:.2f} detik.")
```

1) t0 = time.time(): Catat waktu mulai preprocessing
2) test_df['log'].tolist(): Convert pandas Series ke Python list
3) progress_bar=True: Tampilkan progress bar selama ekstraksi fitur
4) processed_data: DataFrame dengan fitur numerik

- Model Training

```
logging.info("Melatih model...")
t0 = time.time()
detector.train(processed_data)
logging.info(f"Pelatihan model selesai dalam {time.time() - t0:.2f} detik.")
```

1) detector.train(): Method dari AnomalyDetector class
2) Input: Processed numerical features (DataFrame)
3) Unsupervised learning: Tidak menggunakan labels untuk training
4) Performance tracking: Waktu training dimonitor terpisah

- Saving Model and Scaler

```
logging.info("Menyimpan model dan scaler...")
    os.makedirs('models', exist_ok=True)
    detector.save_model('models/anomaly_detector.joblib')
    preprocessor.save_scaler('models/scaler.joblib')
    logging.info("Model dan scaler berhasil disimpan.")
    logging.info(f"Model disimpan ke: models/anomaly_detector.joblib")
    logging.info(f"Scaler disimpan ke: models/scaler.joblib")
    logging.info(f"Total waktu pelatihan: {time.time() - start_time:.2f} detik.")
```

➢ Testing Model

- Import Libraries

```
import pandas as pd
import numpy as np
from preprocessing import LogPreprocessor
from model import AnomalyDetector
from utils import generate_report
import os
import logging
from datetime import datetime
```

- Function generate_anomaly_list()

```
def generate_anomaly_list(logs, predictions, scores, output_path):
    anomaly_indices = np.where(predictions == 1)[0]
```

"np.where(predictions == 1)[0]" berfungsi untuk menemukan indeks semua entri yang diprediksi sebagai anomali

- Create DataFrame

```
anomaly_data = {
        'Index': anomaly_indices + 1,  # 1-based indexing for readability
        'Log_Entry': [logs[idx] for idx in anomaly_indices],
        'Anomaly_Score': [scores[idx] for idx in anomaly_indices]
    }
```

Code ini berfungsi untuk membuat dataframe yang berisi informasi mengenai anomali.

```
df = pd.DataFrame(anomaly_data)
    df = df.sort_values('Anomaly_Score', ascending=False)
```

Code ini berfungsi untuk membuat dataframe dan mengurutkan skor anomali dari terbesar ke terkecil.

```
df.to_csv(output_path, mode='a', index=False)
```

Berfungsi untuk menyimpan dataframe ke dalam file yang sama.

- Funtion test_model()

1. Setup Logging

```
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s [%(levelname)s] %(message)s',
    datefmt='%Y-%m-%d %H:%M:%S'
)
```

Mengkonfigurasi sistem logging dengan format timestamp dan level

2. Loading Test Dataset

```
test_df = pd.read_csv('data/test_logs.csv')
print(f"Total entri pengujian: {len(test_df)}")
print(f"Anomali aktual dalam set pengujian: {test_df['label'].sum()}")
```

    1) Memuat dataset pengujian dari file CSV
    2) Menampilkan statistik dasar dataset

3. Kalkulasi Contamination Rate

```
anomaly_ratio = test_df['label'].mean()
contamination=max(anomaly_ratio, 0.1)
```

1) Menghitung rasio anomali dalam dataset
2) Menggunakan minimum 0.1 sebagai contamination rate untuk model

4. Inisialisasi Components

```
preprocessor = LogPreprocessor()
detector = AnomalyDetector(
    contamination=max(anomaly_ratio, 0.1),
    threshold_percentile=50
)
```

1) Membuat instance preprocessor dan detector
2) Menggunakan contamination rate yang dihitung sebelumnya

5. Loading Tranined Model

```
try:
    detector.load_model('models/anomaly_detector.joblib')
    preprocessor.load_scaler('models/scaler.joblib')
except FileNotFoundError:
    print("Error: File model tidak ditemukan...")
    return
```

1) Memuat model dan scaler yang sudah dilatih
2) Error handling jika file tidak ditemukan

6. Data Processing dan Prediction

```
processed_data = preprocessor.preprocess_logs(test_df['log'].tolist())
predictions = detector.predict(processed_data)
scores = detector.predict_proba(processed_data)
```

1) Memproses data log mentah
2) Membuat prediksi anomali
3) Mendapatkan skor probabilitas anomali

7. Report Generation

```
report_path = os.path.join('reports', 'test_report.txt')
generate_report(test_df['log'].tolist(), predictions, scores, report_path)

anomaly_list_path = os.path.join('reports', 'anomaly_list.csv')
generate_anomaly_list(test_df['log'].tolist(), predictions, scores, anomaly_list_path)
```

1) Menghasilkan laporan teks dan CSV
2) Menyimpan hasil di folder 'reports'

8. Perhitungan Precision dan Recall

```
true_positives = np.sum((true_anomalies == 1) & (predicted_anomalies == 1))
false_positives = np.sum((true_anomalies == 0) & (predicted_anomalies == 1))
false_negatives = np.sum((true_anomalies == 1) & (predicted_anomalies == 0))

precision = true_positives / (true_positives + false_positives)
recall = true_positives / (true_positives + false_negatives)
```

9. Output Hasil

```
print("\nHasil Pengujian:")
print(f"Total log yang dianalisis: {len(test_df)}")
print(f"Anomali aktual dalam dataset: {true_anomalies.sum()}")
print(f"Anomali yang terdeteksi: {predicted_anomalies.sum()}")
print(f"Akurasi: {accuracy:.4f}")
print(f"Presisi: {precision:.4f}")
print(f"Recall: {recall:.4f}")
```

Kode di atas berfungsi untuk menampilkan ringkasan hasil evaluasi model

➢ Visualisasi dan Pelaporan Anomali

- Import Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
import numpy as np
```

- Funtion load_log_data()

```
def load_log_data(file_path):
    with open(file_path, 'r') as f:
        return f.readlines()
```

1) Memuat data log dari file teks dan mengembalikannya sebagai list string.
2) Menggunakan with open() untuk membuka file secara aman (auto-close)
3) Mode 'r' untuk membaca file dalam mode teks
4) f.readlines() membaca seluruh baris dalam file dan mengembalikannya sebagai list
5) Setiap elemen list adalah satu baris log (termasuk karakter newline \n)

- Function plot_anomalies()

```
def plot_anomalies(log_data, predictions, scores, save_path=None):

    plt.figure(figsize=(15, 6))
```

Mengdefinisikan function dan membuat visualisasi dua panel untuk menampilkan skor anomali dan prediksi hasil deteksi. Dan membuat figure matplotlib dengan ukuran 15x6 inch

- Skor Anomali

```
plt.subplot(1, 2, 1)
plt.plot(scores, label='Skor Anomali')
plt.axhline(y=np.mean(scores) + 2*np.std(scores), color='r', linestyle='--', label='Ambang Batas')
plt.title('Skor Anomali')
plt.xlabel('Entri Log')
plt.ylabel('Skor')
plt.legend()
```

1) plt.subplot(1, 2, 1): Membuat subplot pertama dalam grid 1 baris, 2 kolom
2) plt.plot(scores): Membuat line plot skor anomali berdasarkan urutan log
3) plt.axhline(): Menambahkan garis horizontal sebagai threshold
4) np.mean(scores) + 2*np.std(scores): Threshold = rata-rata + 2 standar deviasi
5) Menggunakan aturan statistik "2-sigma" untuk menentukan outlier
6) color='r', linestyle='--': Garis merah putus-putus untuk threshold
7) Label dan legend untuk kemudahan interpretasi

- Prediksi Anomali

```
plt.subplot(1, 2, 2)
plt.scatter(range(len(predictions)), predictions, c=predictions, cmap='coolwarm')
plt.title('Prediksi Anomali')
plt.xlabel('Entri Log')
plt.ylabel('Prediksi (1: Anomali, 0: Normal)')
```

1) plt.subplot(1, 2, 2): Subplot kedua dalam grid yang sama
2) plt.scatter(): Membuat scatter plot untuk prediksi
    a. range(len(predictions)): X-axis adalah index/urutan log
    b. predictions: Y-axis adalah nilai prediksi (0 atau 1)
    c. c=predictions: Warna titik berdasarkan nilai prediksi
    d. cmap='coolwarm': Color map biru-merah (biru untuk normal, merah untuk anomali)

- Function generate_report()

```
def generate_report(log_data, predictions, scores, output_path):
```

1. Ekstraksi Data Anomali

```
def generate_report(log_data, predictions, scores, output_path):

    anomalies = []
    for i, (log, pred, score) in enumerate(zip(log_data, predictions, scores)):
        if pred == 1:  # Anomali
            anomalies.append({
                'index': i,
                'log': log.strip(),
                'score': score
            })
```

1) enumerate(zip(...)): Iterasi dengan index dan gabungan tiga array

2) zip(log_data, predictions, scores): Menggabungkan ketiga array element-wise
3) Kondisi if pred == 1: Hanya memproses log yang diprediksi sebagai anomali
4) log.strip(): Menghilangkan whitespace/newline dari log entry
5) Menyimpan data dalam dictionary dengan struktur yang jelas

2. Sorting Anomali

```
anomalies.sort(key=lambda x: x['score'], reverse=True)
```

1) Mengurutkan anomali berdasarkan skor dari tertinggi ke terendah
2) key=lambda x: x['score']: Menggunakan field 'score' sebagai basis sorting
3) reverse=True: Urutan descending (skor tinggi di atas)

3. Detail Anomali Teratas

```
f.write("Anomali Teratas:\n")
f.write("-" * 80 + "\n")
for anomaly in anomalies[:10]:
    f.write(f"Index: {anomaly['index']}\n")
    f.write(f"Skor: {anomaly['score']:.4f}\n")
    f.write(f"Log: {anomaly['log']}\n")
    f.write("-" * 80 + "\n")
```

1) anomalies[:10]: Slice untuk mengambil 10 anomali teratas saja
2) Format output yang terstruktur dengan separator garis
3) {anomaly['score']:.4f}: Format skor dengan 4 digit desimal
4) Setiap anomali ditampilkan dengan:
    a. Index dalam dataset asli
    b. Skor anomali
    c. Isi log entry yang actual
    d. Garis pemisah untuk readability

D. Hasil Prediksi

Berikut ini adalah hasil laporan dari mendeteksi sebuah data testing yang telah disiapkan sebelumnya.

```
Total log yang dianalisis: 2239541
Jumlah anomali terdeteksi: 1150967

Anomali Teratas:
--------------------------------------------------------------------------------
Index: 76
Skor: 0.7259
Log: 081111 110513 19 INFO dfs.FSNamesystem: BLOCK* ask 10.250.5.237:50010 to delete  blk_6240211623492979493 blk_51818173222797850
--------------------------------------------------------------------------------
Index: 1758
Skor: 0.7259
Log: 081110 103316 19 INFO dfs.FSNamesystem: BLOCK* ask 10.250.14.38:50010 to delete  blk_-223438534269164604 blk_85141115273658588
--------------------------------------------------------------------------------
Index: 2970
Skor: 0.7259
Log: 081110 103416 19 INFO dfs.FSNamesystem: BLOCK* ask 10.251.91.229:50010 to delete  blk_-6027962917485800010 blk_808026542825825
--------------------------------------------------------------------------------
Index: 3416
Skor: 0.7259
Log: 081110 220726 19 INFO dfs.FSNamesystem: BLOCK* ask 10.251.71.193:50010 to delete  blk_-1473834837204851337 blk_-70943122921603
--------------------------------------------------------------------------------
Index: 4523
Skor: 0.7259
Log: 081111 075719 19 INFO dfs.FSNamesystem: BLOCK* ask 10.251.43.115:50010 to delete  blk_6231963138498022919 blk_-550458548339404
--------------------------------------------------------------------------------
```