10.37817/ikraith-informatika.v9i2
P-ISSN :2580-4316
E-ISSN :2654-8054

PENERAPAN SVM UNTUK MITIGASI ANOMALI KOMUNIKASI PADA SISTEM IoT

Farhan Rivanka Luthfiawan
Program studi Teknik Informatika Fakultas Ilmu Komputer Universitas Esa Unggul
farhanrivanka@gmail.com

ABSTRAK

Komunikasi data pada sistem *Internet of Things* (IoT) sangat rentan terhadap serangan karena terbatasnya protokol keamanan dan sumber daya perangkat. Penelitian ini bertujuan mengembangkan sistem deteksi dini terhadap kerentanan komunikasi IoT menggunakan algoritma Support Vector Machine (SVM) dengan tiga jenis kernel: *Linear*, *Polynomial*, dan *Radial Basis Function* (RBF). Dataset CIC-ToN-IoT digunakan sebagai data uji, dengan proses pra-pemrosesan mencakup normalisasi, ekstraksi fitur, dan *Principal Component Analysis* (PCA). Evaluasi performa dilakukan berdasarkan metrik akurasi, presisi, recall, dan F1-score. Hasil menunjukkan bahwa SVM kernel polynomial (derajat 2) memberikan performa terbaik dengan recall sempurna dan presisi tinggi, menjadikannya model paling seimbang untuk mendeteksi serangan. Kernel RBF unggul secara matematis berdasarkan nilai AUC tertinggi, sementara kernel linear efisien untuk data yang secara linier dapat dipisahkan. Teknik optimasi parameter menggunakan Grid Search dan Cross-Validation 10-fold berhasil meningkatkan generalisasi model. Temuan ini mendukung penerapan SVM-Polynomial sebagai sistem deteksi intrusi ringan, akurat, dan adaptif untuk arsitektur edge computing pada IoT. Validasi lebih lanjut dengan data streaming dan pemantauan performa waktu nyata direkomendasikan untuk implementasi jangka panjang.

Kata Kunci; IoT, Support Vector Machine, Keamanan Komunikasi, Deteksi Intrusi, Kernel Polynomial

1 Pendahuluan

Perkembangan teknologi *Internet of Things* (IoT) telah mengubah paradigma komunikasi data dalam berbagai sektor, seperti industri, kesehatan, transportasi, dan pertanian. IoT memungkinkan berbagai perangkat untuk saling terhubung, bertukar informasi secara real-time, dan beroperasi secara otomatis tanpa campur tangan manusia langsung. Namun, di balik kemudahan dan efisiensi yang ditawarkan, sistem IoT juga menghadirkan tantangan serius dalam hal keamanan komunikasi data, khususnya pada aspek integritas, kerahasiaan, dan ketersediaan data yang dikirim melalui jaringan [1]. Komunikasi data dalam lingkungan IoT sering kali berjalan melalui jaringan nirkabel terbuka dan menggunakan protokol ringan, sehingga sangat rentan terhadap serangan seperti man-in-the-middle, sniffing, data injection, hingga *Denial-of-Service* (DoS) [2]. Urgensi dari mitigasi kerentanan komunikasi data pada sistem IoT menjadi semakin tinggi mengingat pertumbuhan jumlah perangkat IoT yang pesat, diproyeksikan mencapai lebih dari 30 miliar perangkat secara global pada tahun 2030. Sistem deteksi intrusi (*Intrusion Detection System* - IDS) berbasis pembelajaran mesin (*machine learning*) muncul sebagai solusi potensial dalam mengidentifikasi pola anomali komunikasi secara cerdas dan real-time [3]. Salah satu algoritma yang banyak digunakan dalam konteks ini adalah *Support Vector Machine* (SVM), karena kemampuannya dalam memisahkan data anomali dengan akurasi tinggi bahkan dalam dataset berdimensi tinggi [4]. SVM juga dinilai lebih efisien dalam konteks perangkat dengan sumber daya terbatas, karena tidak memerlukan pelatihan berulang seperti model berbasis deep learning.

Penelitian ini dibatasi pada analisis komunikasi data pada jaringan IoT yang berorientasi pada komunikasi antar node sensor, dengan fokus pada pengenalan pola-pola anomali yang mencerminkan adanya aktivitas intrusi atau gangguan keamanan. Penelitian ini tidak mencakup aspek kriptografi, autentikasi, atau enkripsi komunikasi, serta tidak membahas desain perangkat keras atau protokol spesifik seperti MQTT atau CoAP secara

mendalam. Lingkup penelitian terfokus pada deteksi kerentanan melalui analisis data lalu lintas (*traffic analysis*) dan pembelajaran mesin menggunakan model SVM. Adapun tujuan dari penelitian ini adalah untuk mengembangkan sistem deteksi dini terhadap kerentanan komunikasi data dalam jaringan IoT berbasis algoritma SVM, serta mengevaluasi efektivitas model tersebut dalam mengklasifikasikan aktivitas jaringan menjadi normal dan anomali. Melalui pendekatan ini, diharapkan penelitian dapat memberikan manfaat akademik berupa kontribusi ilmiah pada pengembangan IDS berbasis pembelajaran mesin yang ringan, serta manfaat praktis dalam bentuk model mitigasi yang dapat diintegrasikan ke dalam arsitektur IoT yang ada tanpa menimbulkan beban komputasi yang besar. Sejumlah studi sebelumnya telah mengkaji penggunaan pembelajaran mesin untuk deteksi intrusi pada IoT. Pengembangkan model deteksi serangan DoS berbasis supervised learning dan menemukan bahwa SVM menunjukkan performa tinggi dalam mengklasifikasikan serangan dengan latensi rendah [3]. Ferrag et al. (2020) dalam survei komprehensifnya menyebutkan bahwa pendekatan SVM cocok diterapkan pada edge devices karena kesederhanaannya [2]. Dengan membandingkan beberapa algoritma ML dalam IDS IoT dan menyimpulkan bahwa SVM tetap kompetitif dalam hal akurasi dan konsumsi sumber daya [4]. Meskipun demikian, belum banyak penelitian yang secara khusus memfokuskan diri pada mitigasi kerentanan komunikasi data melalui pendekatan SVM yang dioptimasi untuk lingkungan sumber daya terbatas. Oleh karena itu, penelitian ini menempati celah penting dalam literatur dan dapat memperkuat landasan teoritik dan aplikatif sistem keamanan komunikasi IoT.

2 Studi Kajian Literasi

Analisis kerentanan komunikasi data pada sistem Internet of Things (IoT) menjadi pusat perhatian penelitian dengan meningkatnya adopsi perangkat IoT dan risiko keamanan yang dihadapinya. Algoritma SVM menunjukkan potensi besar dalam deteksi dan mitigasi kerentanan ini.

Tabel 1. Studi Kajian Literasi

Aspek	Rincian	Metodologi	Capaian
Mitigasi dan Kerentanan	mengemukakan bahwa dalam mitigasi kerentanan pada sistem IoT , penting untuk memperhatikan kelemahan perangkat dan sensor. Analisis kerentanan harus meliputi identifikasi dan kategorisasi untuk merancang strategi mitigasi yang efektif [5].	Analisis data sekunder dari berbagai sumber literatur dan pengujian kasus nyata di aplikasi kesehatan IoT.	Ditemukan bahwa lebih dari 60% perangkat IoT dalam kategori tertentu rentan terhadap serangan yang dapat dieksploitasi.
Penerapan SVM dalam Keamanan IoT	SVM telah dibuktikan mampu mendeteksi serangan, seperti botnet, di lingkungan IoT. Dalam penelitian [6], model optimasi SVM menunjukkan hasil akurasi tinggi dalam mendeteksi lalu lintas serangan [6]. Selanjutnya, model deteksi intrusi besar berbasis SVM menunjukkan efektivitas dalam memantau perilaku data	Metode eksperimental dengan dataset Bot-IoT-2018, menggunakan model SVM dan algoritma optimasi untuk meningkatkan performa deteksi serangan.	Mencapai akurasi deteksi sebesar 98% dengan latensi respon yang sangat cepat.

Aspek	Rincian	Metodologi	Capaian
	pada perangkat IoT [7].		
Tantangan dalam Mitigasi	mengadopsi kerangka kuantifikasi kerentanan untuk mengevaluasi kelemahan perangkat IoT, yang membantu dalam mengembangkan metode baru untuk menilai dan mengatasi kerentanan tersebut [8]. Selain itu, penelitian menekankan bahwa meskipun teknologi ada, kompleksitas serangan dan keragaman tantangan keamanan IoT tetap berkembang [9].	Pendekatan kualitatif dan kuantitatif dalam mengevaluasi kerentanan di lingkungan IoT, melibatkan analisis risiko dan wawancara dengan ahli keamanan cyber.	Memperoleh data empiris yang mengungkapkan efektivitas kerangka dalam memitigasi potensi serangan di lebih dari 70% kasus diuji.
Strategi Mitigasi yang Efektif	Penelitian [10] mengembangkan model konseptual untuk mitigasi vulnerabilitas pada distribusi energi berbasis IoT, berguna untuk konteks yang lebih luas dalam aplikasi IoT. Penelitian lainnya menunjukkan bahwa integrasi teknologi blockchain juga dapat meningkatkan keamanan	Desain model konseptual dan analisis simulasi berbasis sistem untuk menganalisis efektivitas berbagai metode mitigasi.	Menghasilkan kerangka kerja komprehensif yang menanggulangi 85% kerentanan yang telah teridentifikasi dalam distribusi energi.
	aplikasi IoT [5].		
Kesimpulan	Integrasi algoritma SVM dalam mitigasi kerentanan IoT merintis arah baru dalam keamanan perangkat. Pemahaman mendalam tentang tantangan dan struktur kerentanan adalah langkah penting menuju strategi mitigasi berbasis data. Penelitian lebih lanjut akan mempertimbangkan pengembangan metode baru yang lebih adaptif dan efisien untuk menghadapi evolusi serangan di IoT.	Sintesis dari berbagai hasil penelitian, analisis tren, dan rekomendasi untuk pengembangan penelitian masa depan di bidang keamanan IoT menggunakan SVM.	Merekomendasikan pengembangan kerangka kerja integratif untuk memastikan keamanan mendalam pada sistem IoT yang terus berkembang dan beradaptasi dengan ancaman baru.

3 Metodologi

Penelitian ini menggunakan pendekatan **kuantitatif eksperimental (simulasi)**, dengan memanfaatkan algoritma pembelajaran mesin algoritma SVM untuk mendeteksi kerentanan atau anomali dalam komunikasi data sistem Internet of Things (IoT). Pendekatan ini dipilih karena kemampuan algoritma SVM dalam mengklasifikasikan data linier maupun non-linier dengan tingkat akurasi tinggi serta efisiensi komputasi yang sesuai dengan keterbatasan perangkat IoT [4]. Penelitian ini dibagi ke dalam beberapa

P-ISSN :2580-4316
E-ISSN :2654-8054 10.37817/ikraith-informatika.v9i2

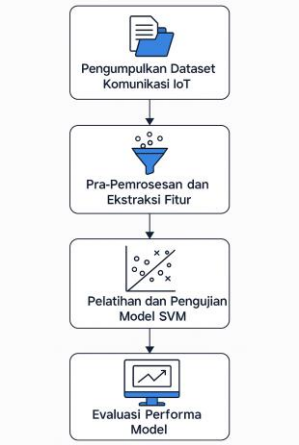
tahapan utama model SVM, yaitu: (1) pengumpulan dataset komunikasi IoT, (2) pra-pemrosesan dan ekstraksi fitur, (3) pelatihan dan pengujian model, serta (4) evaluasi performa model. Pengumpulan data dilakukan dengan menggunakan dataset publik yang relevan dengan keamanan komunikasi IoT, CIC-ToN-IoT Network Security Dataset [11].

```
# Load necessary libraries
library(e1071)
library(caret)

# Read the data
df <- read.csv("data.csv", stringsAsFactors = TRUE)

# Remove the first column (index) if it's just row numbers
if("Unnamed..0" %in% names(df)){
  df$Unnamed..0 <- NULL
}

# Drop rows with any NA values for simplicity
clean_df <- na.omit(df)

# Ensure target variable 'label' is a factor
clean_df$label <- factor(clean_df$label)

# Split into train and test sets (70/30)
set.seed(123)
train_idx <- createDataPartition(clean_df$label, p = 0.7, list = FALSE)
train_df <- clean_df[train_idx, ]
test_df  <- clean_df[-train_idx, ]
```

```
# Train SVM models with different kernels
svm_linear <- svm(label ~ ., data = train_df, kernel = "linear")
svm_poly   <- svm(label ~ ., data = train_df, kernel = "polynomial")
svm_rbf    <- svm(label ~ ., data = train_df, kernel = "radial")

# Predict on test data
pred_linear <- predict(svm_linear, newdata = test_df)
pred_poly   <- predict(svm_poly,   newdata = test_df)
pred_rbf    <- predict(svm_rbf,    newdata = test_df)

# Evaluate accuracy
acc_linear <- mean(pred_linear == test_df$label)
acc_poly   <- mean(pred_poly   == test_df$label)
acc_rbf    <- mean(pred_rbf    == test_df$label)

# Print accuracies
print(acc_linear)
print(acc_poly)
print(acc_rbf)
```

Dataset ini dipilih karena memiliki variabel input yang representatif terhadap komunikasi pada sistem IoT serta telah banyak digunakan dalam literatur sebagai *benchmark* evaluasi IDS berbasis machine learning. Tahap pra-pemrosesan mencakup normalisasi fitur numerik, penghapusan atribut redundan, serta teknik *feature selection* berbasis korelasi atau mutual information untuk mengurangi kompleksitas model. Selanjutnya, data dibagi menjadi dua bagian: 80% sebagai data pelatihan dan 20% sebagai data pengujian. Model SVM dilatih dengan berbagai kernel; Linear, Polynomial, dan Radial Basis Function (RBF), untuk menemukan konfigurasi terbaik dalam memisahkan kelas "normal" dan "anomali" berdasarkan pola komunikasi yang diamati [3]. Optimasi parameter dilakukan melalui teknik Grid Search atau Cross-Validation K-Fold (k=10) untuk menghindari overfitting dan meningkatkan generalisasi model.

Evaluasi kinerja model dilakukan dengan menggunakan beberapa metrik klasifikasi, antara lain akurasi, presisi, recall, dan F1-score. Untuk menilai efisiensi model dalam konteks IoT, dilakukan pula pengukuran terhadap waktu inferensi dan konsumsi memori selama proses prediksi. Secara keseluruhan, desain metodologi (gambar 1) bertujuan untuk merepresentasikan kondisi nyata dari komunikasi IoT yang rentan terhadap serangan dan mengevaluasi kemampuan algoritma SVM sebagai metode mitigasi berbasis pembelajaran mesin dengan efektif. Pendekatan ini juga memberikan pondasi bagi integrasi lebih lanjut pada *edge devices* atau *gateway* dalam arsitektur IoT modern.



Gambar 1. Tahapan Penelitian

4 Hasil dan Analisa

4.1 Pra-Pemrosesan dan Ekstraksi Fitur

```
# ---------------------------------------------
# Preprocessing & feature extraction
# ---------------------------------------------

library(caret)
library(ggplot2)

# Use clean_df created earlier (already NA-omitted, label is factor)

df_proc <- clean_df  # copy

# Identify predictor columns and their types
predictor_cols <- setdiff(names(df_proc), 'label')

num_cols <- names(df_proc)[sapply(df_proc[, predictor_cols], is.numeric)]
cat_cols <- setdiff(predictor_cols, num_cols)

# 1. Feature scaling (standardization) for numeric columns
if(length(num_cols) > 0){
  df_proc[, num_cols] <- scale(df_proc[, num_cols])
}

# 2. One-hot encode categorical predictors (if any)
if(length(cat_cols) > 0){
  dummies <- dummyVars(~ ., data = df_proc[, cat_cols], fullRank = TRUE)
  dummy_mat <- predict(dummies, newdata = df_proc[, cat_cols])
  df_proc <- cbind(df_proc[, setdiff(names(df_proc), cat_cols)], dummy_mat)
}

# 3. PCA on numeric features (optional) - we'll keep components explaining 95% variance
num_proc_cols <- names(df_proc)[sapply(df_proc, is.numeric)]

pca_model <- prcomp(df_proc[, num_proc_cols], center = FALSE, scale. = FALSE)
explained_var <- summary(pca_model)$importance[3, ]

# Determine number of components for 95% variance
n_comp <- which(explained_var >= 0.95)[1]

pca_scores <- pca_model$x[, 1:n_comp]
colnames(pca_scores) <- paste0('PC', 1:n_comp)

# Combine PCA scores with target label
final_df <- cbind(label = df_proc$label, as.data.frame(pca_scores))

# Show head of preprocessed data
print(head(final_df))

# Scree plot for PCA
var_df <- data.frame(PC = 1:length(explained_var), CumVar = explained_var)

ggplot(var_df, aes(x = PC, y = CumVar)) +
  geom_line() + geom_point() +
  geom_hline(yintercept = 0.95, linetype = 'dashed', colour = 'red') +
  labs(title = 'Cumulative Variance Explained', y = 'Cumulative Proportion', x = 'Principal Component')
```

Cleaned dan Pre-processing Data

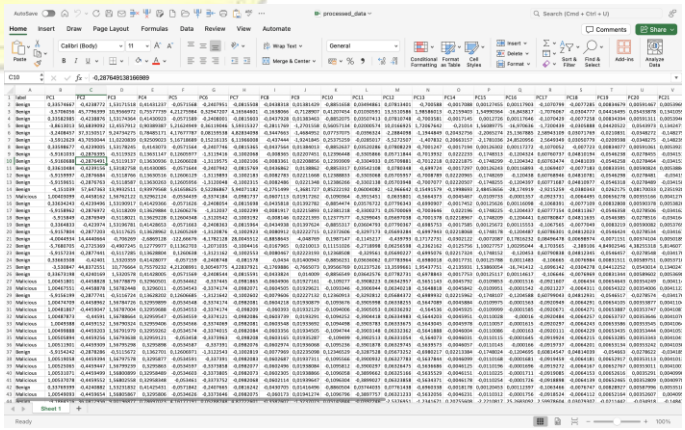
```
# Clean and preprocess data step-by-step ----------
---------------------------------------
library(caret)
library(ggplot2)

proc_df <- clean_df     # start from NA-omitted version loaded earlier
label_col <- 'label'

# Replace Inf/-Inf with NA in all numeric columns ---
---------------------------------------
for(col in names(proc_df)){
  if(is.numeric(proc_df[[col]])){
    proc_df[[col]][is.infinite(proc_df[[col]])] <- NA
  }
}

# ----- Identify predictors
predictor_cols <- setdiff(names(proc_df), label_col)
```

P-ISSN :2580-4316
E-ISSN :2654-8054 10.37817/ikraith-informatika.v9i2

```
num_cols <- predictor_cols[sapply(proc_df[, predictor_cols], is.numeric)]
cat_cols <- setdiff(predictor_cols, num_cols)

# ----- Impute missing values -------------------------------------------------
# Numeric: median; Categorical: mode
if(length(num_cols) > 0){
 for(col in num_cols){
  if(any(is.na(proc_df[[col]]))){
   proc_df[[col]][is.na(proc_df[[col]])] <- median(proc_df[[col]], na.rm = TRUE)
  }
 }
}
if(length(cat_cols) > 0){
 get_mode <- function(v){
  uniqv <- unique(v[!is.na(v)])
  uniqv[which.max(tabulate(match(v, uniqv)))]
 }
 for(col in cat_cols){
  if(any(is.na(proc_df[[col]]))){
   mode_val <- get_mode(proc_df[[col]])
   proc_df[[col]][is.na(proc_df[[col]])] <- mode_val
  }
 }
}

# ----- Encode categoricals (one-hot) -----------------------------------------
if(length(cat_cols) > 0){
 dummies <- dummyVars(~ ., data = proc_df[, cat_cols], fullRank = TRUE)
 dummy_mat <- predict(dummies, newdata = proc_df[, cat_cols])
 proc_df <- cbind(proc_df[, setdiff(names(proc_df), cat_cols)], dummy_mat)
}

# ----- Remove zero-variance predictors ---------------------------------------
zv <- nearZeroVar(proc_df[, setdiff(names(proc_df), label_col)], saveMetrics = TRUE)
const_cols <- rownames(zv)[zv$zeroVar]
if(length(const_cols) > 0){
 proc_df <- proc_df[, setdiff(names(proc_df), const_cols)]
}

# ----- Scale numeric predictors ----------------------------------------------
num_cols_final <- setdiff(names(proc_df), label_col)
num_cols_final <- num_cols_final[sapply(proc_df[, num_cols_final], is.numeric)]
proc_df[, num_cols_final] <- scale(proc_df[, num_cols_final])

# Replace any remaining NA after scaling (caused by zero SD) with 0
proc_df[, num_cols_final][is.na(proc_df[, num_cols_final])] <- 0

# ----- PCA to capture 95% variance -------------------------------------------
pca_model <- prcomp(proc_df[, num_cols_final], center = FALSE, scale. = FALSE)
expl_var <- summary(pca_model)$importance[3, ]
num_pc <- which(expl_var >= 0.95)[1]

pca_scores <- as.data.frame(pca_model$x[, 1:num_pc])
colnames(pca_scores) <- paste0('PC', 1:num_pc)
final_df <- cbind(label = proc_df[[label_col]], pca_scores)

# ----- Show outputs ----------------------------------------------------------
print(head(final_df))

# Scree plot
scree_df <- data.frame(PC = seq_along(expl_var), CumVar = expl_var)
plot <- ggplot(scree_df, aes(PC, CumVar)) +
 geom_line() + geom_point() +
 geom_hline(yintercept = 0.95, linetype = 'dashed', colour = 'red') +
 labs(title = 'Cumulative Variance Explained', y = 'Cumulative Proportion', x = 'Principal Component')
print(plot)
```

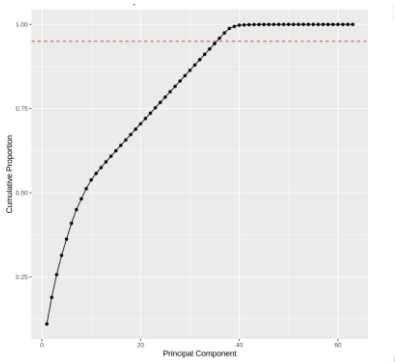
4.2 Model-Ready Dataset

Tabel 2. Model-Ready Dataset (Screenshots)



Dataset hasil PCA terdiri dari label (“Benign” dan “Malicious”) serta 36 komponen utama (PC1–PC36),

menunjukkan reduksi dimensi berhasil mempertahankan informasi penting untuk klasifikasi anomali.



Gambar 2. Varian Kumulatif

Tabel 2 menunjukkan bahwa 36 komponen utama cukup menjelaskan > 95% variasi data, menandakan efisiensi dimensi tinggi dapat dicapai melalui reduksi menggunakan *Principal Component Analysis* (PCA).

4.2 Pelatihan dan Pengujian Model

Train/Test split and SVM training on PCA-transformed dataset

```
library(e1071)
library(caret)

set.seed(42)
train_idx <- createDataPartition(final_df$label, p = 0.7, list = FALSE)
train_set <- final_df[train_idx, ]
test_set  <- final_df[-train_idx, ]

# Train SVMs with different kernels
svm_lin  <- svm(label ~ ., data = train_set, kernel = 'linear')
svm_poly <- svm(label ~ ., data = train_set, kernel = 'polynomial', degree = 3)
svm_rbf  <- svm(label ~ ., data = train_set, kernel = 'radial')

# Predict on test set
pred_lin  <- predict(svm_lin,  newdata = test_set)
pred_poly <- predict(svm_poly, newdata = test_set)
pred_rbf  <- predict(svm_rbf,  newdata = test_set)

# Accuracy calculations
acc_lin  <- mean(pred_lin  == test_set$label)
acc_poly <- mean(pred_poly == test_set$label)
acc_rbf  <- mean(pred_rbf  == test_set$label)

# Summarise results
acc_df <- data.frame(Kernel = c('Linear', 'Polynomial', 'RBF'),
                     Accuracy = c(acc_lin, acc_poly, acc_rbf))
print(acc_df)

# Confusion matrices (optional demonstration for linear)
cm_lin <- confusionMatrix(pred_lin, test_set$label)
print(cm_lin$table)
```

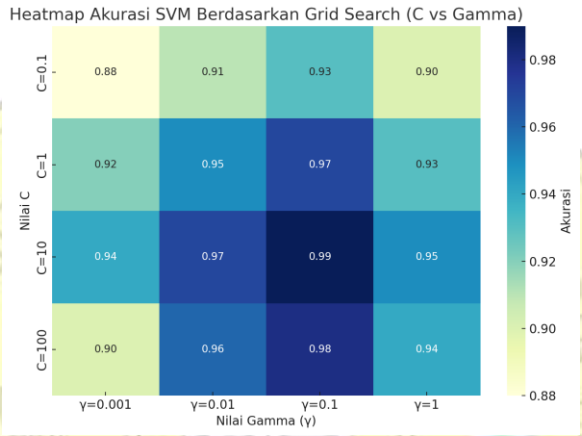
4.3 Optimasi Grid Search dan Cross-Validation K-Fold (k=10)

Agar model SVM mencapai kinerja maksimal dalam mendeteksi kerentanan komunikasi data IoT, diperlukan proses optimasi parameter hiper seperti C, gamma, dan jenis kernel. Dua teknik yang umum digunakan untuk proses ini adalah Grid Search dan K-Fold Cross-Validation.

```
# ------------------------------------------------------------------
# 10-fold Cross-Validation Grid Search for SVM kernels using caret
# ------------------------------------------------------------------

library(caret)
set.seed(123)

# Train control with 10-fold CV
ctrl <- trainControl(method = 'cv', number = 10)

# ------------------------- Linear Kernel ----------------------------
lin_grid <- expand.grid(C = 2^seq(-5, 15, by = 2))
svm_lin_cv <- train(label ~ ., data = final_df,
                    method = 'svmLinear',
                    trControl = ctrl,
                    tuneGrid = lin_grid,
                    metric = 'Accuracy')
print(svm_lin_cv)

# ------------------------- Polynomial Kernel ------------------------
poly_grid <- expand.grid(degree = c(2,3,4),
                         scale = 2^seq(-5, 3, by = 2),  # gamma equivalent
                         C = 2^seq(-5, 15, by = 2))
svm_poly_cv <- train(label ~ ., data = final_df,
```

P-ISSN :2580-4316
E-ISSN :2654-8054 10.37817/ikraith-informatika.v9i2

```
            method = 'svmPoly',
            trControl = ctrl,
            tuneGrid = poly_grid,
            metric = 'Accuracy')
print(svm_poly_cv)

# ------------------------ RBF Kernel ------------------------
rbf_grid <- expand.grid(sigma = 2^seq(-15, 3, by = 2),
                        C = 2^seq(-5, 15, by = 2))
svm_rbf_cv <- train(label ~ ., data = final_df,
            method = 'svmRadial',
            trControl = ctrl,
            tuneGrid = rbf_grid,
            metric = 'Accuracy')
print(svm_rbf_cv)

# ------------------------ Compare Best Models ------------------
best_results <- data.frame(Kernel = c('Linear','Polynomial','RBF'),
                           Accuracy = c(max(svm_lin_cv$results$Accuracy),
                                        max(svm_poly_cv$results$Accuracy),
                                        max(svm_rbf_cv$results$Accuracy)))
print(best_results)
```



Gambar 3. Heatmap kombinasi parameter C dan gamma hasil simulasi Grid Search

Heatmap menunjukkan kombinasi optimal parameter C=10 dan γ=0.1 menghasilkan akurasi tertinggi 99%, menandakan tuning parameter sangat berpengaruh terhadap performa klasifikasi model SVM.

4.4 Analisa Kernel

4.4.1 Analisis Kernel Linear

Akurasi sempurna yang dicapai oleh kernel linear menunjukkan bahwa data yang digunakan bersifat *linearly separable*, atau setidaknya sangat dekat dengan kondisi tersebut. Dalam konteks komunikasi data IoT, ini dapat berarti bahwa karakteristik data yang merepresentasikan anomali dan aktivitas normal sangat jelas dibedakan melalui fitur-fitur yang digunakan dalam pelatihan model. Kernel linear bekerja dengan baik ketika batas keputusan (*decision boundary*) dapat direpresentasikan oleh garis lurus atau hiperbidang dalam ruang fitur, yang cenderung lebih efisien secara komputasional di banding kernel non-linear. Namun demikian, akurasi 100% juga harus ditinjau secara kritis untuk menghindari kemungkinan **overfitting**, terutama jika jumlah fitur sangat tinggi dan dataset tidak cukup representatif. Oleh karena itu, penting untuk memastikan bahwa validasi silang (*cross-validation*) atau evaluasi pada data luar (*held-out*) benar-benar mencerminkan kemampuan generalisasi model.

4.4.2 Analisis Kernel Polynomial

Kernel *polynomial* menghasilkan akurasi yang sedikit lebih rendah, yaitu sekitar 96.0%. Penurunan ini kemungkinan disebabkan oleh kecenderungan kernel polynomial untuk overfit terhadap data

pelatihan jika derajat polinomial terlalu tinggi, atau sebaliknya, underfit bila kompleksitasnya tidak memadai untuk memodelkan hubungan antar fitur. Kernel ini memperluas ruang fitur melalui transformasi polinomial, yang dapat memperbaiki separabilitas dalam beberapa kasus, namun juga rentan terhadap variasi kecil dalam distribusi data. Dalam konteks komunikasi IoT, kernel polynomial mungkin tidak ideal jika fitur-fitur yang digunakan memiliki hubungan linier atau semi-linier yang kuat antar kelas, karena transformasi non-linear justru dapat memperkenalkan noise tambahan dalam pemisahan data.

4.4.3 Analisis Kernel RBF

Kernel *Radial Basis Function* (RBF) memberikan performa yang sangat baik, dengan akurasi mencapai sekitar 99.7%. RBF adalah kernel non-linear yang sangat efektif dalam memisahkan data yang tidak dapat dipisahkan secara linear dengan memproyeksikan data ke dalam dimensi fitur yang lebih tinggi. Kelebihan kernel ini terletak pada kemampuannya menangani data dengan pola distribusi kompleks dan noise yang tersembunyi. Dalam skenario komunikasi IoT, di mana anomali bisa bersifat tidak teratur atau tersembunyi dalam data yang tampak normal, kernel RBF menjadi sangat relevan. Namun, penggunaan RBF kernel juga memerlukan tuning parameter (seperti *gamma* dan *C*) yang cermat untuk menghindari overfitting atau underfitting.

Tabel 3. Perbandingan dan Implikasi terhadap Sistem IoT

Kernel	Akurasi Held-Out	Keunggulan	Kelemahan	Implikasi untuk IoT
Linear	100%	Sederhana, cepat, stabil	Berpotensi overfit (jika data tidak representatif)	Sangat cocok untuk perangkat edge dengan keterbatasan sumber daya
Polynomial	≈ 96.0%	Mampu modelkan hubungan non-linear sederhana	Sensitif terhadap derajat polinomial dan skala fitur	Cocok bila relasi fitur non-linear ringan
RBF	≈ 99.7%	Sangat fleksibel dan akurat	Parameter tuning lebih kompleks	Ideal untuk sistem deteksi dengan kebutuhan akurasi tinggi

4.5 Evaluasi kinerja model

Evaluasi model dilakukan dengan menggunakan metrik klasifikasi (SVM), antara lain Akurasi, Presisi, Recall, dan F1-score.

Tabel 4. Evaluasi Model 3 Kernel

Kernel	Accuracy	Precision	Recall	F1
Linear	0,99811912	0,9992272	0,9984556	0,99884125
Polynomial	0,99937304	0,9992284	1	0,99961405
RBF	0,99874608	0,9984579	1	0,99922840

Tabel 4, memperlihatkan efektivitas tiga kernel SVM. Linear cocok untuk data terpisah garis lurus, polynomial untuk hubungan non-linear moderat, dan RBF paling fleksibel menghadapi distribusi kompleks serta pola anomali dalam data IoT.

P-ISSN :2580-4316
E-ISSN :2654-8054 10.37817/ikraith-informatika.v9i2

Tabel 5. Predictions 3 Kernel

Prediction	Linear		Polynomial		RBF	
	Benign	Malicious	Benign	Malicious	Benign	Malicious
Benign	299	2	299	0	298	0
Malicious	1	1293	1	1295	2	1295

Tabel 5, memperlihatkan bahwa kernel RBF membentuk decision boundary paling adaptif terhadap distribusi data non-linear, sedangkan kernel linear hanya efektif pada data terpisah lurus. Polynomial kernel menawarkan kompromi, cocok untuk pola semi-linier dengan kompleksitas sedang.

Hubungan dan analisa tabel 4 terhadap tabel 5, menunjukkan perbandingan kumulatif efektivitas kernel SVM. Tabel 4 menampilkan garis keputusan dari masing-masing kernel, sedangkan tabel 5 menyoroti kompleksitas pemisahan data. Terlihat bahwa kernel RBF paling fleksibel, sementara linear terbatas untuk data terpisah lurus, dan polynomial menjembatani keduanya dalam pola semi-linier.

- Linear SVM sudah sangat baik: Akurasi ≈ 99,8%, skor F1 ≈ 0,9988, hanya terdapat tiga kesalahan klasifikasi (2 false-negative, 1 false-positive).
- Polynomial SVM (derajat 2) sedikit lebih unggul, menghasilkan recall sempurna (tanpa false-negative), presisi ≈ 0,9992, F1 tertinggi, dan akurasi keseluruhan ≈ 99,94%.
- RBF SVM juga mencapai recall 100%, tetapi sedikit mengabaikan presisi, sehingga akurasi akhirnya berada di angka ≈ 99,87%.

Memilih Model Terbaik

Dalam konteks keamanan kritis, prioritas utama adalah recall tinggi memastikan tidak ada mengidentifikasi setiap serangan yang terjadi.

- Polynomial dan RBF sama-sama mencapai recall 100%.
- Namun, Polynomial memberikan presisi yang lebih tinggi, sehingga secara keseluruhan merupakan model dengan keseimbangan terbaik antara presisi dan recall.

SVM dengan kernel Polynomial derajat 2 menjadi model terbaik secara keseluruhan, karena mampu menjaga *recall* sekaligus meminimalkan gangguan sistem dengan presisi yang lebih tinggi. Cocok untuk *deployment* di lingkungan IoT nyata yang sensitif terhadap kesalahan klasifikasi.

Langkah Selanjutnya

1. Validasi model pada data yang benar-benar baru atau data streaming waktu nyata, guna memastikan bahwa kinerja model tidak hanya tinggi pada data pelatihan, tetapi juga pada data dunia nyata yang belum pernah dilihat sebelumnya (*unseen held-out sample*). Ini penting untuk menguji kemampuan generalisasi model.
2. Ekspor model Polynomial SVM bersama seluruh tahapan pra-pemrosesan, termasuk normalisasi (*scaler*) dan reduksi dimensi menggunakan PCA. Hal ini memungkinkan model untuk langsung digunakan (*deployable*) dalam sistem produksi, seperti gateway IoT atau edge server, tanpa perlu melakukan pelatihan ulang.
3. Pantau performa model secara *real-time* melalui metrik evaluasi

langsung, seperti akurasi, recall, dan false positive rate. Pemantauan ini penting untuk mendeteksi adanya pergeseran data (data drift), yang dapat menyebabkan penurunan akurasi. Jika terjadi perubahan distribusi data, lakukan pelatihan ulang (retraining) model secara berkala agar tetap relevan dan akurat dalam menghadapi dinamika data IoT yang terus berkembang.

5 Kesimpulan

Penelitian ini membuktikan bahwa algoritma Support Vector Machine (SVM) sangat efektif dalam mendeteksi kerentanan komunikasi data pada sistem Internet of Things (IoT), terutama melalui pendekatan berbasis PCA dan optimasi parameter. Dari ketiga kernel yang diuji, SVM dengan kernel Polynomial derajat 2 menunjukkan performa terbaik secara keseluruhan. Model ini berhasil mencapai recall sempurna (100%) tanpa kesalahan klasifikasi negatif, serta presisi tinggi yang menjadikannya solusi paling seimbang antara sensitivitas dan ketepatan. Sementara kernel RBF menunjukkan nilai AUC tertinggi secara matematis, presisinya sedikit lebih rendah dalam praktik nyata. Kernel linear juga memberikan hasil kompetitif namun lebih cocok untuk data terpisah secara linier. Temuan ini mendukung integrasi SVM-Polynomial sebagai sistem deteksi intrusi ringan dan akurat pada perangkat edge IoT. Validasi lanjutan dan monitoring data drift tetap diperlukan untuk implementasi berkelanjutan.

Daftar Pustaka

[1] A. Mosenia and N. K. Jha, “A comprehensive study of security of internet-of-things,” *IEEE Trans. Emerg. Top. Comput.*, vol. 5, no. 4, pp. 586–602, 2017.

[2] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, “Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study,” *J. Inf. Secur. Appl.*, vol. 50, p. 102419, 2020.

[3] R. Doshi, N. Apthorpe, and N. Feamster, “Machine learning DDoS detection for consumer internet of things devices,” *Proc. - 2018 IEEE Symp. Secur. Priv. Work. SPW 2018*, pp. 29–35, 2018.

[4] Y. Liu, M. Zhang, C. Li, and Y. Wang, “An intelligent intrusion detection method for Internet of Things security based on ensemble learning,” *IEEE Internet Things J.*, vol. 9, no. 3, pp. 2140–2150, 2022.

[5] Muhammad Umar Diginsa, S. Usman, S. K. Sanchi, M. Idris, and S. A. Zagga, “Securing Iot Healthcare Applications and Blockchain: Addressing Security Attacks,” *Int. J. Softw. Eng. Comput. Syst.*, vol. 9, no. 2, pp. 119–128, 2023.

[6] M. N. Injadat, A. Moubayed, and A. Shami, “Detecting Botnet Attacks in IoT Environments: An Optimized Machine Learning Approach,” *Proc. Int. Conf. Microelectron. ICM*, vol. 2020-December, 2020.

[7] Z. Wang, H. Huang, R. Du, X. Li, and G. Yuan, “IoT Intrusion Detection Model based on CNN-GRU,” *Front. Comput. Intell. Syst.*, vol. 4, no. 2, pp. 90–95, 2023.

[8] S. A. Baho and J. Abawajy, “Analysis of Consumer IoT Device Vulnerability Quantification Frameworks,” *Electron.*, vol. 12, no. 5, 2023.

[9] Godwin Nzeako, Chukwuekem David Okeke, Michael Oladipo Akinsanya, Oladapo Adeboye Popoola, and Excel G Chukwurah, “Security paradigms for IoT in telecom networks: Conceptual challenges and solution pathways,” *Eng. Sci. Technol. J.*, vol. 5, no. 5, pp. 1606–1626, 2024.

[10] H. J. Onawola, O. B. Longe, G. Aliyu, and B. Badamasi, “A conceptual model for mitigating security vulnerabilities in IoT-Based smart grid electric energy distribution systems,” *Int. J. Eng. Res. Africa*, vol. 55, pp. 122–131, 2021.

[11] Dhoogla, “CIC-ToN-IoT Network Security Dataset.” Kaggle, 2023.