Penerapan Algoritma *Support Vector Machine* (SVM) Dalam Memprediksi *Employee Turnover*

SKRIPSI

Diajukan untuk memenuhi salah satu syarat

memperoleh gelar Sarjana Komputer Program

Studi Informatika

Disusun oleh:

AMADEUS PALI HAMAPATI

195314084

PROGRAN STUDI INFORMATIKA FAKULTAS SAINS DAN TEKNOLOGI UNIVERSITAS SANATA DHARMA YOGYAKARTA

2024

Application of Support Vector Machine (SVM) Algorithm in Predicting Employee Turnover

THESIS

Present as partial Fulfillment of the requirement To

obtain the *Sarjana Komputer* Degree

In informatics Study Program

Created By:

AMADEUS PALI HAMAPATI

195314084

INFORMATICS STUDY PROGRAM FACULTY OF

SCIENCE AND TECHNOLOGY SANATA DHARMA

UNIVERSITY YOGYAKARTA

2024

HALAMAN PENGESAHAAN

SKRIPSI

PenerapanAlgoritma*SupportVectorMachine*(SVM) Dalam Memprediksi *Employee Turnover*

Disusun
oleh:
Amadeus
Pali
Hamapati

195314084

Telah Disetujui Oleh :

Dosen Pembimbing,

Dr. Cyprianus Kuntoro Adi, S.J. MA., MSc. Tanggal : 10 juni 2024

HALAMAN PERSETUJUAN

ABSTRAK

Penelitian ini bertujuan untuk menerapkan algoritma Support Vector Machine (SVM) dalam memprediksi tingkat employee turnover di sebuah perusahaan menggunakan dataset dari situs Kaggle. Employee turnover merujuk pada rasio jumlah karyawan yang meninggalkan industri dalam periode tertentu, dibagi dengan jumlah rata-rata karyawan selama periode tersebut. Proses penelitian meliputi analisis data awal, preprocessing untuk membersihkan dan normalisasi data, serta transformasi data menggunakan Principal Component Analysis (PCA). Seleksi fitur dilakukan dengan metode recursive feature elimination (RFE), dengan mempertimbangkan variasi nilai 'k'. Data diimbangi menggunakan Synthetic Minority Over sampling Technique (SMOTE), sementara variasi parameter dilakukan pada kernel (RBF, Linear, dan Sigmoid), nilai C (0.1, 1.0, 10), dan Gamma (0.1, 1.0, 10), dievaluasidengan nilai k-fold 3, 5, 7, 9. Penelitian juga mencoba variasi dari 3, 5, 7, dan 9 fitur.

Hasil eksperimen menunjukkan bahwa penggunaan kernel 'RBF', nilai C '10', dan gamma '10' dengan nilai k=9 dan 9 fitur memberikan akurasi tertinggi sebesar 84.72%. Kesimpulannya, kombinasi parameter kernel 'RBF', nilai C '10', dan Gamma '10' memberikan hasil optimal dengan penggunaan nilai k-fold 9 dan 9 fitur. Saran untuk penelitian berikutnya mencakup penggunaan algoritma selain SVM, penambahan variasi parameter, dan penambahan GUI

Kata Kunci : *Data Mining, Klasifikasi, SVM, K-Fold, Employee turnover*

ABSTRACT

This study aims to implement the Support Vector Machine (SVM) algorithm in predicting the rate of employee turnover in a company using a dataset from the Kaggle website. Employee turnover refers to the ratio of the number of employees leaving the industry within a certain period, divided by the average number of employees during that period. The research process includes initial data analysis, preprocessing to clean and normalize data, and data transformation using Principal Component Analysis (PCA). Feature selection is carried out using the recursive feature elimination (RFE) method, considering various values of 'k'. Data is balanced using the Synthetic Minority Over sampling Technique (SMOTE), while parameter variations are performed on kernels (RBF, Linear, and Sigmoid), C values (0.1, 1.0, 10), and Gamma values (0.1, 1.0, 10), evaluated with k-fold values of 3, 5, 7, 9. The study also explores variations of 3, 5, 7, and 9 features.

The experimental results show that using the 'RBF' kernel, C value of '10', and gamma value of '10' with k=9 and 9 features yields the highest accuracy of 84.72%. In conclusion, the combination of 'RBF' kernel, C value of '10', and Gamma value of '10' provides optimal results with the use of a k-fold value of 9 and 9 features. Recommendations for future research include the use of algorithms other than SVM, adding parameter variations, and incorporating a GUI.

Keywords: Data Mining, Classification, SVM, K-Fold, Employee turnover

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas berkat dan penyertaan-Nya, sehingga penulis dapat menyelesaikan skripsi dengan judul "Penerapan Algoritma *Support Vector Machine*(SVM) Dalam Memprediksi *Employee Turnover*" dengan baik adanya. Terselesaikannya skripsi ini tak lepas dari dukungan, bimbingan, nasehat dan doa dari berbagai pihak. Penulis mengucapkan terima kasih kepada:

1. Orang tua, Bapak Hinggu panjanji dan Ibu Agustina Rija Hamapati serta kakak AdikYohanes hapu Amah dan Angela Rambu Ndohi yang senantiasa mendukung, memotivasi, dan mendoakan penulis dalam menyelesaikan skripsi.
2. Romo Cyprianus Kuntoro Adi, S.J. M.A., M.Sc., Ph.D., selaku dosen pembimbing skripsi yang telah bersedia meluangkan waktu untuk membimbing dan memberikan masukan kepada penulis selama proses penyusunan skripsi.
3. Bapak Ir. Drs. Haris Sriwindono, M.Kom, Ph.D., selaku Dekan Fakultas Sains dan Teknologi Universitas Sanata Dharma.
4. Bapak Bapak Dr. Ir. Iwan Binanto., selaku Ketua Program Studi Informatika Universitas Sanata Dharma.
5. Seluruh dosen program studi Informatika Universitas Sanata Dharma yang telah memberikan pengalaman dan pengetahuan selama berkuliah.
6. Ignas, Joy, Carles, Mario, Nando, Cesar, Leo, Pedro, Samuel, Ridvan,Jo dan Daffa, selaku teman terdekat yang selalu disamping dalam proses penulisan skripsi ini. Dukungan, obrolan ringan dan semangat bersama telah membuat proses ini menjadi lebih menyenangkan.

7. Terimakasih untuk teman KKN Feby,Anggi ,Lidya,Boni,Nia dan Mega yang menemani saya selama ini.
8. Terima kasih kepada seluruh anggota Teksapala atas dukungan dan inspirasi yang telah diberikan selama ini.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Employee turnover, atau pergantian karyawan, merupakan rasio jumlah karyawan yang meninggalkan industri sepanjang periode tertentu, dipecah dengan jumlah rata-rata karyawan sepanjang periode tersebut [1]. Akhir-akhir ini, *employee turnover* telah menjadi pembicaraan hangat dalam dunia bisnis sejak tahun 2019. Rata-rata, tingkat *employee turnover* bertambah sebesar 8,7%, dan diperkirakan akan bertambah lebih besar pada tahun 2023, dengan diperkirakan tingkat keluar-masuk karyawan sebesar 35,6% di Inggris serta 46,8% di Amerika Serikat [3]. Akibatnya tidak hanya terbatas pada keberlanjutan bisnis, namun juga dapat meningkatkan tingkat *turnover* di berbagai zona industri.

Employee turnover yang tinggi adalah masalah umum yang dapat memengaruhi kinerja dan pertumbuhan perusahaan [3]. Dengan jumlah *turnover* yang tinggi, dapat juga menyebabkan kerugian terhadap perusahaan karena proses perekrutan, seleksi, dan pelatihan karyawan baru dapat menjadi biaya yang signifikan bagi perusahaan. Oleh sebab itu, kemampuan untuk memprediksi *employee turnover* akan menjadi alat yang sangat berharga bagi setiap perusahaan yang ingin mempertahankan karyawan dan memprediksi perilaku masa depan mereka. Tidak hanya mengurangi biaya, mengatur *turnover* juga membantu industri mempertahankan area kerja yang normal serta produktif.

Prediksi *employee turnover* menjadi elemen krusial dalam pengambilan keputusan strategis. Ini membolehkan industri untuk mengenali pola serta faktor-faktor yang berkontribusi pada *employee turnover*. Dalam konteks ini, algoritma *Support Vector Machines (SVM)* timbul sebagai pemecahan potensial. Bebrapa

penelitian menemukan hasil jika *SVM* merupakan salah satu algoritma *Machine Learning* terbaik yang biasanya digunakan untuk pengenalan pola [3].

Support vector machine (SVM) adalah algoritma *supervised machine learning (mempunyai label)* yang digunakan untuk tugas klasifikasi dan regresi [2]. *Support vector machine* merupakan salah satu algoritma klasifikasi dan regresi yang *powerful* dalam berbagai bidang aplikasi, serta salah satu algoritma yang terbaik dalam mengoptimalkan solusi yang diinginkan [2]. Algoritma *SVM* sudah banyak diterapkan dalam penelitian seperti kategorisasi teks dan *hiper teks, image classification, bioinformatika, hand-written character recognition, face detection*, dan lainnya [2], hal tersebut menunjukkan bahwa algoritma *SVM* dapat digunakan dalam banyak hal termasuk dalam memprediksi tingkat *employee turnover*. Salah satu fitur sangat menonjol dari *SVM* adalah kemampuannya untuk mengoptimalkan margin antara kelas-kelas informasi dalam tugas klasifikasi. *SVM* sangat sesuai untuk riset ini karena bisa membangun model prediktif yang dapat mengenali pola-pola terkait dengan *employee turnover*. Algoritma *SVM* memiliki kelebihan dalam menanggulangi informasi yang tidak linier secara efektif, yang cocok dengan kompleksitas ikatan antara faktor-faktor dalam topik riset ini. Dengan memakai *SVM*, industri bisa meningkatkan model prediktif yang kokoh untuk mengenali faktor-faktor signifikan dalam *employee turnover*, membantu pengambilan langkah-langkah penangkalan yang cocok untuk meningkatkan retensi karyawan serta kesejahteraan industri secara totalitas.

1.2. Rumusan Masalah

Dari hasil penelitian pada dataset Employee Turnover, penulis tertarik untuk mengimplementasikan algoritma Support Vector Machines (SVM) untuk prediksi turnover karyawan, dengan pertanyaan-pertanyaan berikut:

1. Bagaimana pemodelan terbaik algoritma Support Vector Machines (SVM) dalam prediksi tingkat Employee Turnover? Penelitian ini akan mengeksplorasi

berbagai jenis kernel SVM dan menguji berbagai parameter SVM untuk menemukan model yang paling sesuai dengan dataset Employee Turnover yang digunakan.

2. Sejauh mana tingkat akurasi yang dapat dicapai oleh model terbaik Support Vector Machines (SVM) dalam prediksi Employee Turnover?

1.3. Batasan masalah

Dalam penelitian ini terdapat batasan masalah agar penelitian ini tidak terlalu luas. Batasan masalah yang ada dalam penelitian ini adalah sebagai berikut:

1. data set yang digunkan di ambil dari kaggle
2. data set merupakan data karyawan yang keluar dari perusahaan selama rentang waktu 2016-2020.
3. algoritma yang digunakan hanya satu yaitu SVM

1.4. Manfaat penelitian

Pembaca akan mendapatkan manfaat dari penelitian ini, manfaat tersebut antara lain:

1. Menambah pemahaman tentang aplikasi algoritma SVM dalam memprediksi employee turnover.
2. Memberikan dasar untuk mengembangkan strategi pencegahan turnover yang efektif.
3. Manbah ilmu baru tentang pengaplikasian SVM dalam konteks manajemen sumber daya manusia.

1.5. Tujuan penelitian

Tujuan penelitian ini dapat dilihat dalam beberapa poin berikut:

1. Melihat kemampuan algoritma SVM dalam memeprediksi employee turnover. penelitian ini bertujuan untuk mengevaluasi seberapa efektif algoritma SVM dalam memprediksi tingkat turnover karyawan berdasarkan data yang tersedia.

2. Mencari kombinasi parameter terbaik dari algoritma SVM dalam memprediksi employee turnover. Penelitian ini bertujuan untuk menemukan kombinasi parameter terbaik (kernel, C, dan gamma) yang menghasilkan akurasi tertinggi dalam memprediksi employee turnover.

BAB II

TINJAUAN PUSTAKA DAN LANDASAN TEORI

Dalam bab ini akan dijelaskan tentang dasar teori yang digunakan sebagai acuan dalam melakukan penelitian ini . Bab ini antara lain akan menjelaskan tentang tinjauan pustaka yang berisi penelitian yang menjadi acuan penelitian ini,pada bab ini juga akan membahas materi pendukung penelitian ini mulai dari pengertian *Machine Learning*,klasifikasi,*SVM*,*Feature extraction*, *K-fold Cross Validation* dan *Confusion Matrix.*

2.1 Tinjauan Pustaka

Penelitian ini terinspirasi dari penelitian-penelitian terdahulu yang berkaitan dengan *support vector machine* ataupun *Employee trunover*. Penelitian-penelitian tersebut menjadi tambahan referensi bagi penulis dalam pengembangan penelitian ini.

Penelitian pertama, Prediksi *Employee Attrition* menggunakan Algoritma *Support Vector Machine* , tujuan utama dari penelitian ini adalah Membandingkan performansi algoritma SVM dengan algoritma kNN pada permasalahan prediksi employee attrition

.Hasil terbaik yang bidapat daari penelitian ini adalah menunjukkan bahwa model dengan algoritma SVM memiliki nilai metrik yang lebih baik dari pada algoritma kNN dengan rata-rata *accuracy* 0.86, *F1-score* 0.59, dan *geometric-mean* 0.75.

Penelitian lainnya, tentang perbandingan beberapa algoritma Klasifikasi. Pada penelitian ini membandingkan algorima *Naïve Bayes, K-Nn Dan Svm.* dalam penelitian ini di temukan bahwa metode SVM lebih baik daripada Naïve Bayes dan K-NN. Dengan menggunakan 33 data uji, SVM memiliki akurasi 84,9%, presisi 85,1%, sedangkan K-NN memiliki akurasi 81,8%, presisi 84,1%, dan Naïve Bayes memiliki akurasi 78,8% dan presisi 80,1%.

Penelitian ketiga, Efektivitas SVM dalam Deteksi Penyakit Kanker dari Data Genomik**,** bertujuan untuk mengevaluasi kinerja SVM dalam mendeteksi jenis kanker dari data genomik. Hasil menunjukkan bahwa SVM memiliki akurasi 90%, lebih tinggi dibandingkan dengan Random Forest yang memiliki akurasi 87%, menegaskan keunggulan SVM dalam aplikasi deteksi kanker dari data genomik.

Penelitian keempat, Prediksi Keberhasilan Kampanye Pemasaran Menggunakan SVM**,** bertujuan untuk memprediksi hasil kampanye pemasaran menggunakan SVM dan Decision Tree. Hasil menunjukkan bahwa algoritma SVM menghasilkan akurasi 82% dan F1-score 0.76, sedangkan Decision Tree menghasilkan akurasi 78% dan F1-score 0.70. Ini menunjukkan keunggulan SVM dalam memprediksi keberhasilan kampanye pemasaran.

Penelitian kelima, Klasifikasi Teks Berita dengan Menggunakan SVM**,** bertujuan untuk mengklasifikasikan berita ke dalam kategori tertentu menggunakan SVM dan Logistic Regression. Hasil menunjukkan bahwa SVM mencapai akurasi 88% dan presisi 0.85, lebih baik dibandingkan dengan Logistic Regression yang mencapai akurasi 85% dan presisi 0.82. Ini menunjukkan keunggulan SVM dalam klasifikasi teks berita.

Uraian tentang penelitian terdahulu terangkum pada **Tabel 2.1** berikut.

Tabel 2. 1Penelitian Terdahulu

Peneliti	**Topik Penelitian**	**Algoritma**	**Hasil**
Muhammad Abdurrohman Al Fatih (2023)	Prediksi Employee Attrition menggunakan	Support Vector Machine & k-Nearest Neighbors	Hasil terbaik adalah algoritma SVM dengan

	Algoritma Support Vector Machine (SVM)		rata-rata accuracy 0.86, F1-score 0.59, dan geometric-mean 0.75
Novendra Adisaputra Sinaga1, B. Herawan Hayadi , Zakarias Situmorang (2022)	Comparison of machine learning algorithms for clinical event prediction (risk of coronary heart disease)	Perbandingan Akurasi Algoritma Naïve Bayes, K-Nn Dan Svm Dalam Memprediksi Penerimaan Pegawa	Dengan menggunakan 33 data uji, SVM memiliki akurasi 84,9%, presisi 85,1%, sedangkan K-NN memiliki akurasi 81,8%, presisi 84,1%, dan Naïve Bayes memiliki akurasi 78,8% dan presisi 80,1%.
Laura Balzano, Robert Nowak (2020)	Efektivitas SVM dalam Deteksi Penyakit Kanker dari Data Genomik	Support Vector Machine	SVM menunjukkan akurasi 90% dalam mendeteksi jenis kanker tertentu dari data genomik, dibandingkan dengan 87% yang diperoleh dari

			Algoritma Random Forest.
ohn Doe, Jane Smith (2019)	Prediksi Keberhasilan Kampanye Pemasaran Menggunakan SVM	Support Vector Machine & Decision Tree	Algoritma SVM menghasilkan akurasi 82% dan F1-score 0.76, sementara Decision Tree memiliki akurasi 78% dan F1-score 0.70, menunjukkan keunggulan SVM dalam memprediksi hasil kampanye pemasaran.
Ahmed Patel, Maria Gonzalez (2021)	Klasifikasi Teks Berita dengan Menggunakan SVM	Support Vector Machine & Logistic Regression	SVM mencapai akurasi 88% dan presisi 0.85 dalam klasifikasi berita ke dalam kategori tertentu, lebih baik dibandingkan dengan Logistic Regression yang mencapai akurasi 85% dan presisi 0.82.

2.2 Landasan teori

2.2.1 Machine Learning (ML)

Machine Learning (ML), atau *Pembelajaran Mesin*, merupakan bidang ilmu komputer yang mengkaji cara membuat program yang mampu menghasilkan pengetahuan baru dan menggunakan pengetahuan yang sudah ada, yang sering disebut sebagai *pengalaman* atau *data*. Ini berbeda dengan pengetahuan yang secara langsung diprogram ke dalam program. Istilah yang lebih umum mengacu pada kemampuan membuat komputer belajar dari lingkungan sekitarnya sehingga dapat mengembangkan "pengetahuan" yang berkembang seiring waktu.

Dalam *ML*, terdapat dua teknik dasar utama, yaitu *Supervised Learning* dan *Unsupervised Learning*. *Supervised Learning* adalah jenis algoritma *ML* di mana model belajar dari informasi berupa data yang telah diberi label. Dengan kata lain, algoritma ini dapat mengenali pola dan menghasilkan output berdasarkan data yang telah dipelajari sebelumnya. Terdapat dua tipe utama dalam *Supervised Learning*, yaitu *regresi* dan *klasifikasi* [5].

1. *Regresi*: Jenis *Supervised Learning* yang bertujuan untuk memprediksi nilai numerik atau kontinu. Algoritma *regresi* membangun model untuk menghubungkan variabel input dengan nilai output yang bersifat *comtinue*.
2. *Klasifikasi*: Jenis *Supervised Learning* yang fokus pada pemberian label pada data berdasarkan kelas atau kelompok tertentu. Algoritma *klasifikasi* mempelajari pola dari data yang diberi label dan kemudian dapat memberikan prediksi label untuk data baru.

Dengan adanya teknik *Supervised Learning*, *ML* dapat diaplikasikan dalam berbagai konteks, termasuk prediksi, klasifikasi, dan analisis pola dalam data. Selain itu, *ML* memungkinkan pembangunan model matematis yang dapat digunakan untuk mengumpulkan pengetahuan dan data, seperti yang terlihat pada contoh sederhana prediksi kata di perangkat seluler atau pengenalan wajah di platform seperti *Facebook*.

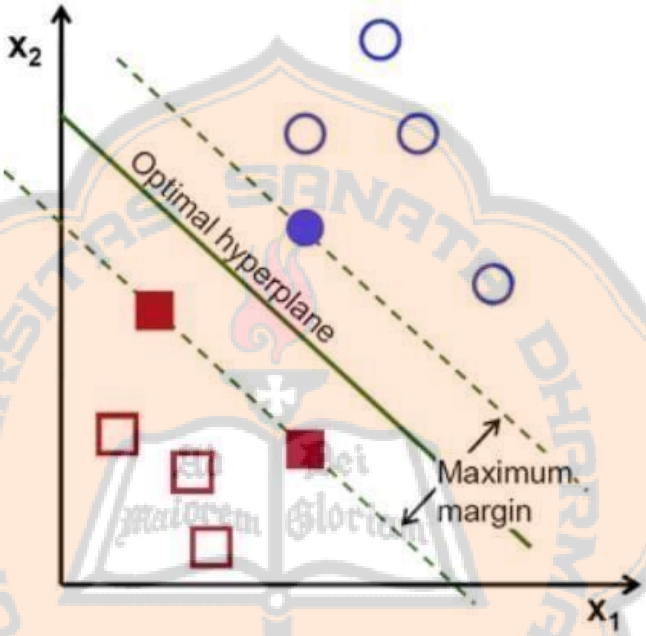
2.2.2 Klasifikasi

Klasifikasi adalah teknik pembelajaran mesin yang bersifat supervisi di mana model berusaha untuk memprediksi label yang tepat berdasarkan data masukan yang diberikan. Proses klasifikasi melibatkan pelatihan penuh model menggunakan dataset pelatihan, yang kemudian diuji pada dataset pengujian sebelum diaplikasikan untuk membuat prediksi pada data baru yang belum pernah dilihat sebelumnya [6].

Teknik klasifikasi umumnya digunakan untuk memprediksi atau menjelaskan dataset yang memiliki kategori biner atau nominal. Namun, teknik klasifikasi menjadi kurang efektif ketika berurusan dengan kategori ordinal, seperti ketika mencoba mengklasifikasikan seseorang ke dalam kelompok dengan tingkat pendapatan tinggi, sedang, atau rendah. Hal ini disebabkan karena teknik klasifikasi tidak memperhatikan urutan implisit di antara kategori-kategori tersebut. Selain itu, hubungan lain seperti subclass-superclass antara kategori, misalnya, hubungan antara manusia dan siamang sebagai primata yang merupakan subclass dari mamalia, sering diabaikan oleh teknik klasifikasi.[7]Contoh algoritma *klasifikasi* meliputi *Support Vector Machines (SVM)*, *Decision Trees*, *Random Forest*, *Logistic Regression*, dan banyak lagi. Pemilihan algoritma tergantung pada karakteristik data, kompleksitas masalah, dan persyaratan spesifik dari tugas *klasifikasi* yang dihadapi.

2.2.3 Support Vector Machine (SVM)

Support Vector Machine (SVM) adalah algoritma supervised yang digunakan untuk klasifikasi dan regresi. Prinsip dasar kerja SVM melibatkan pencarian hyperplane terbaik yang memisahkan kelas dengan margin maksimal, yaitu jarak antara data terdekat dari masing-masing kelas[2].



Gambar 2. 1 hyperparameter

Dalam konteks klasifikasi, data yang diberikan biasanya terdiri dari pasangan fitur dan label kelas. SVM berusaha menemukan hyperplane dalam ruang fitur berdimensi tinggi yang memisahkan data ke dalam dua kelas dengan cara yang paling optimal. Hyperplane ini adalah garis pemisah optimal antara kedua kelas, dan margin adalah jarak dari hyperplane ke data terdekat dari masing-masing kelas, yang dikenal sebagai support vector[2]. Data yang berada paling dekat dengan hyperplane disebut sebagai support vector karena mereka "mendukung" atau menentukan posisi hyperplane.

Secara matematis, hyperplane dalam ruang n-dimensi dapat didefinisikan oleh persamaan berikut: $w \cdot x + b = 0 w \cdot x + b = 0$ dimana:

- w adalah vektor bobot,
- x adalah vektor fitur,
- b adalah bias.

Klasifikasi dilakukan dengan menentukan sisi mana dari hyperplane sebuah titik data berada. Titik data diklasifikasikan ke dalam kelas positif jika memenuhi:

$w \cdot xi + b \geq 1 w \cdot xi + b \geq 1$ dan ke dalam kelas negatif jika memenuhi: $w \cdot xi + b \leq -1 w \cdot xi + b \leq -1$

a) Maksimalisasi Margin

Untuk memaksimalkan margin, yaitu jarak antara hyperplane dan support vector, SVM memformulasikan masalah ini sebagai masalah optimasi. Tujuannya adalah untuk meminimalkan

$||w||2 ||w||2$ dengan syarat: $yi(w \cdot xi + b) \geq 1 yi(w \cdot xi + b) \geq 1$ dimana yi adalah label kelas dari data ke-i.

Masalah ini kemudian dapat diselesaikan menggunakan teknik optimasi seperti *Lagrange Multipliers*, yang menghasilkan solusi optimal untuk parameter w dan bb.

b) Penggunaan Fungsi Kernel

Dalam banyak kasus, data tidak dapat dipisahkan secara linear dalam ruang fitur aslinya. Untuk mengatasi masalah ini, SVM menggunakan fungsi

kernel untuk memetakan data ke ruang fitur berdimensi lebih tinggi di mana data dapat dipisahkan secara linear. Fungsi kernel memungkinkan SVM untuk menemukan hyperplane optimal dalam ruang fitur yang lebih tinggi tanpa harus secara eksplisit menghitung koordinat dari data dalam ruang tersebut.

Berikut adalah beberapa fungsi kernel yang umum digunakan dalam SVM:

1. Karnel Linear : Kernel linear digunakan ketika data dapat dipisahkan secara linear dalam ruang fitur aslinya.

$$\text{Linear kernel: } K(xi, xj) = (xi, xj) \quad (2.1)$$

2. Karnel Polynomial: Kernel polinomial digunakan untuk memetakan data ke dalam ruang fitur berdimensi lebih tinggi menggunakan polinomial. Parameter y mengontrol skala input, r adalah koefisien bebas, dan dd adalah derajat polinomial.

$$\text{Polynomial kernel: } K(xi, xj) = (y(xi, xj) + r)d \quad (2.2)$$

3. Karnel RBF : Kernel RBF, juga dikenal sebagai Gaussian kernel, digunakan untuk memetakan data ke dalam ruang fitur berdimensi tak hingga. Ini sangat efektif untuk menangani data yang tidak terpisahkan secara linear. Parameter y mengontrol jarak pengaruh dari satu titik data, sementara C adalah parameter regulasi yang mengontrol trade-off antara mencapai margin maksimal dan mengklasifikasikan data dengan benar

$$\text{RBF kernel: } K(xi, xj) = \exp(-y||xj-xj||2+ C) \quad (2.3)$$

4. Karnel Sigmoid: Kernel sigmoid digunakan dalam jaringan saraf tiruan dan mengimplementasikan fungsi aktivasi sigmoid. Parameter y dan r mengontrol bentuk dan skala dari fungsi kernel ini.

Sigmoid kernel: K (xi, xj) = tanh (y(xi , xj)+ r) (2.4)

. Keterangan, y: *gamma;* C: *cost*; r =*coefficient*; d = *degree.*

c) Hyperparameter dalam SVM

Dalam penerapan fungsi kernel, beberapa hyperparameter perlu ditentukan dengan hati-hati untuk mendapatkan kinerja terbaik dari model SVM. Hyperparameter yang sering digunakan adalah cost (C) dan gamma (y).

- **Cost (C):** Parameter ini mengontrol kesalahan dan menghindari misklasifikasi pada data pelatihan. Nilai C yang tinggi berarti model berusaha mengklasifikasikan semua data pelatihan dengan benar, yang dapat menyebabkan overfitting. Sebaliknya, nilai C yang rendah memungkinkan beberapa data pelatihan untuk salah klasifikasi, yang dapat meningkatkan generalisasi model.
- **Gamma (y):** Parameter ini mengatur jarak pengaruh dari satu titik data training**.** Semakin tinggi nilai gamma, dampaknya menjadi lebih terlokalisasi, sehingga model cenderung menjadi lebih kompleks. Sebaliknya, nilai gamma yang rendah membuat model lebih sederhana dan lebih mungkin untuk menangkap pola global dalam data.

2.2.4 Employee Turnover

employee turnover adalah mengukur jumlah karyawan yang meninggalkan sebuah organisasi dalam jangka waktu tertentu, biasanya dalam satu tahun atau lebih[11].e*mployee turnover* mencakup karyawan yang meninggalkan posisi secara sukarela dan mereka yang dipisahkan secara paksa dari perusahaan melalui pemutusan hubungan kerja, pengurangan pegawai, atau pemecatan, apakah peran tersebut diisi kembali oleh orang lain atau tidak. Tingkat employee turnover adalah cara untuk mengukur seberapa sering karyawan meninggalkan sebuah perusahaan dan digantikan oleh yang baru. Ini dihitung dengan membagi jumlah karyawan yang meninggalkan selama periode tertentu dengan jumlah rata-rata karyawan di perusahaan selama waktu yang sama[11]. Tingkat perputaran biasanya diungkapkan sebagai persentase.*employee turnover* adalah hal yang alami untuk setiap organisasi. Namun, perputaran yang tinggi dapat menjadi tantangan karena sebuah perusahaan menghabiskan waktu dan uang untuk mencari dan melatih karyawan baru[11].

2.2.5 Feature Extraction

Feature Extraction adalah proses transformasi data mentah menjadi fitur numerik untuk diproses, dengan tetap mempertahankan informasi dalam kumpulan data asli. Pendekatan ini menghasilkan kinerja yang lebih baik dibandingkan dengan penerapan pembelajaran mesin langsung pada data mentah[8]. Terdapat dua metode ekstraksi fitur, yaitu manual dan otomatis. Metode manual melibatkan identifikasi dan deskripsi fitur yang relevan, memerlukan pemahaman yang baik tentang latar belakang atau domain untuk membuat keputusan informasional. Seiring berjalannya waktu, insinyur dan ilmuwan telah mengembangkan metode ekstraksi fitur untuk berbagai jenis data

seperti gambar, sinyal, dan teks.[8] Di sisi lain, *Feature Extraction* otomatis menggunakan algoritma khusus atau jaringan mendalam untuk mengekstrak fitur secara otomatis dari sinyal atau gambar tanpa campur tangan manusia. Teknik ini sangat berguna ketika kecepatan beralih dari data mentah ke pengembangan algoritma pembelajaran mesin diperlukan. Salah satu contoh ekstraksi fitur otomatis adalah gelombang sebaran (*wavelet scattering*).[8] Pada penelitian ini, digunakan pendekatan ekstraksi fitur otomatis, dengan fokus pada metode Analisis Komponen Utama (PCA).

2.2.5.1 Analisis Komponen Utama (PCA)

PCA adalah metode yang berguna dalam ekstraksi fitur yang bertujuan untuk mengurangi dimensi data dengan mempertahankan informasi yang paling penting. Dengan mentransformasi ruang fitur asli ke ruang yang lebih rendah dimensinya, yang disebut sebagai ruang komponen utama, PCA menemukan vektor-vektor eigen dari matriks kovarian data, merepresentasikan arah di mana data memiliki variasi maksimum. Dengan merangkum variasi data ke dalam beberapa komponen utama, PCA memungkinkan kita untuk bekerja dengan representasi fitur yang lebih sederhana dan ringkas. Ini membantu mengatasi masalah high-dimensional data, mencegah overfitting, dan meningkatkan efisiensi komputasi. Meskipun PCA secara otomatis mengekstrak fitur dari data, interpretasi hasilnya masih memerlukan pemahaman yang kuat tentang konsep aljabar linier dan statistik.

2.2.6 Feature Selection

Feature selection adalah cara untuk menyederhanakan data dengan memilih hanya beberapa fitur penting sambil menghilangkan data yang tidak berguna, berulang, atau mengganggu. Ini biasanya akan meningkatkan kinerja model dengan meningkatkan akurasi, mengurangi biaya komputasi, dan membuat model lebih mudah dimengerti.Secara sederhana, fitur yang tidak

relevan adalah yang tidak membantu memisahkan data dari berbagai kelas atau kelompok. Menghapusnya tidak akan merusak pembelajaran; bahkan, bisa membantu proses pembelajaran dengan menghindari kebingungan dan pemborosan sumber daya [10]. Untuk klasifikasi dengan sampel pelatihan terbatas dan dimensi tinggi, pemilihan fitur memainkan peran penting dalam menghindari masalah *overfitting* dan meningkatkan kinerja klasifikasi. Salah satu metode pemilihan fitur yang umum digunakan untuk masalah sampel terbatas adalah metode *recursive feature elimination (RFE)*. Metode *RFE* memanfaatkan kemampuan generalisasi yang tertanam dalam support vector machines dan oleh karena itu cocok untuk masalah dengan sampel terbatas. Meskipun kinerjanya baik, *RFE* cenderung mengabaikan fitur-fitur "lemah" yang mungkin memberikan peningkatan kinerja yang signifikan ketika dikombinasikan dengan fitur lainnya.[10]

2.2.7 K-fold Cross Validation

K-fold Cross Validation adalah teknik evaluasi performa model yang membagi dataset menjadi k subset (*fold*) dan menggunakan k-1 subset untuk melatih model, sementara satu subset digunakan untuk pengujian. metode ini umum digunakan untuk evaluasi model. Ini adalah jenis validasi silang yang digunakan untuk mengukur kinerja suatu algoritma dengan membagi sampel data secara acak menjadi k kelompok, di mana salah satu kelompok menjadi data pengujian dan sisanya menjadi data pelatihan. Jumlah k dapat bervariasi, seperti 3, 5, 7, 9, dan lainnya. Pada gambar 2.1, visualisasi cara kerja k-fold cross validation dengan k=3 akan diberikan.

K = 3		
data validasi	data testing	data testing
data testing	data validasi	data testing
data testing	data testing	data validasi

Gambar 2. 2 Visualisasi cara kerja cross validation

2.2.8 Confusion Matrix

Confusion Matrix adalah tabel yang digunakan untuk mengukur performa model klasifikasi. Ini menyajikan jumlah data yang diklasifikasikan dengan benar (true positives dan true negatives) serta yang diklasifikasikan dengan kesalahan (false positives dan false negatives). Penjelasan akan mencakup bagaimana Confusion Matrix digunakan untuk menghitung metrik evaluasi seperti akurasi, presisi, recall, dan F1-score.

Tabel 2. 2Confusion matriks

confusion matriks		nilai sebenarnya	
		True	False
hasil prediksi	True	TP	FP
	False	FN	TN

Ket:

TP (true positif) : benar memprediksi nilai yang benar

TN (true negatif) : benar memprediksi nilai yang salah

FP (false positif) : salah memprediksi nilai yang benar

FN (false positif) : salah memprediksi nilai yang salah

1. Akurasi

   Akurasi (*accurancy*) disini adalah matriks untuk menghitung seberapa akurat algoritma SVM.

   rumus:

$$\frac{TP+TN}{TP+TN+FP+TN} \quad (2.5)$$

2. Presisi

   Presisi (*precision*) disini adalah matriks untuk menghitung seberapa presisi algoritma SVM.

   rumus:

$$\frac{TP}{TP+FP} \quad (2.6)$$

3. Recall

   Recall disini adalah matriks untuk menghitung seberapa presisi model yang digunakan untuk mengklasifikasi data.

   rumus:

$$\frac{TP}{TP+FN} \quad (2.7)$$

4. F1-Score

F1-Score disini adalah mengidentifikasi model klasifikasi dari *precision dan recall*. Nilai terbaik dari *f1-score* adalah 1.0 sedangkan nilai terburuknya adalah 0. Jika *F1-score* mendapatkan hasil yang baik berarti model klasifikasi antara presisi dan *recall* baik.

rumus:

$$F1\ Score = 2\ \mathrm{X}\frac{\text{Presisi X Recall}}{\text{Presisi}+\text{Recall}} \quad (2.8)$$

BAB III

METODOLOGI PENELITIAN

Dalam bab ini peneliti menguraikan metodologi penelitian yang digunakan untuk memprediksi *employee turnover* dengan algoritma *SVM*.Bab tiga ini dimulai pengenalan data ,Skenario penelitian ,*preprocesing,Feature Extraction,balancing,split data,modoeling* dan *confuxion matrix* dan skenario pengujian.

3.1 Dataset

Penelitian ini menggunakan data *Employee turnover* yang di ambil dari Kaggle. Data set ini terdiri dari 10 kolom dan 9540 baris. Data ini melibatkan informasi seperti *department, promoted, review, projects, salary, tenure, satisfaction, bonus, avg_hrs_month*, dan memiliki kolom sebagai label yang bernama *left* yang menyatakan apakah karyawan yang bersangkutan itu keluar dari perusahaan atau tidak,kolom ini bernilai *yes* dan *no* ,diamana *yes* dinyatakan meninggalkan perusahaan,*no* diyatakann tinggal. data set ini termasuk data yang *imbalanced* di karenakan ada 2784 di kelas *yes* dan 6756 dikelas *no*. berikut penjelasan dari setiap fitur

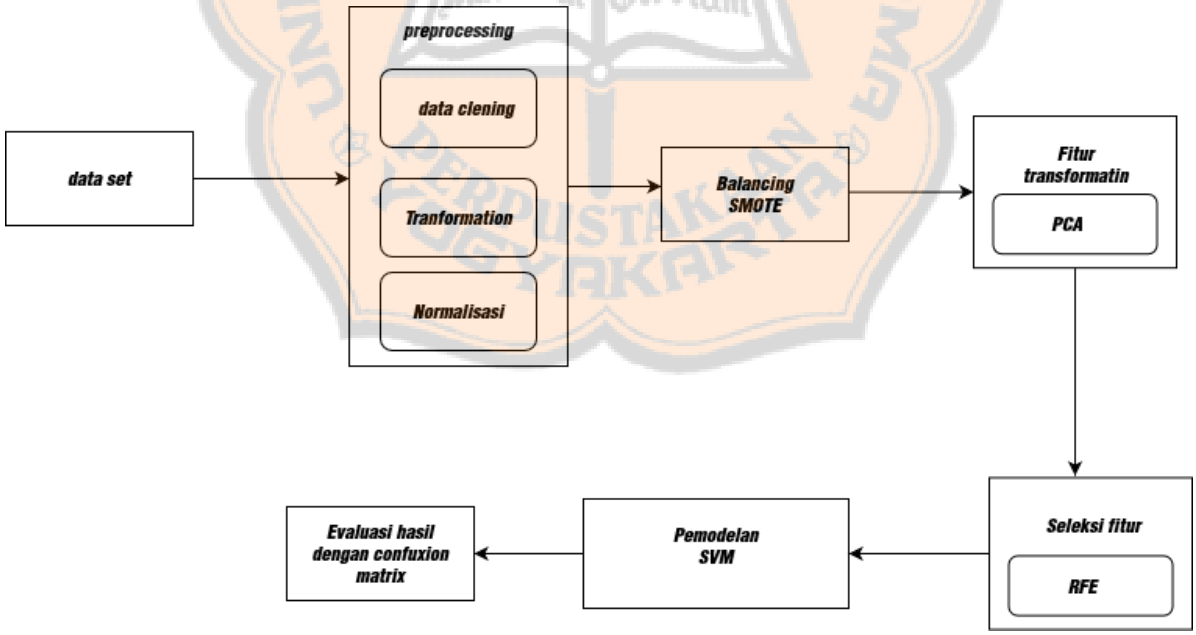
1. "department" - departemen tempat karyawan bekerja.
2. "promoted" - 1 jika karyawan dipromosikan dalam 24 bulan terakhir, 0 jika tidak.
3. "review" - skor gabungan yang diberikan kepada karyawan dalam evaluasi terakhir mereka.
4. "projects" - jumlah proyek yang diikuti oleh karyawan.
5. "salary" - karena alasan kerahasiaan, gaji dibagi menjadi tiga tingkat: rendah, menengah, tinggi.
6. "tenure" - berapa lama karyawan telah bekerja di perusahaan tersebut.
7. "satisfaction" - ukuran kepuasan karyawan dari survei.

8. "bonus" - 1 jika karyawan menerima bonus dalam 24 bulan terakhir, 0 jika tidak.
9. "avg_hrs_month" - rata-rata jam kerja karyawan dalam sebulan.
10. "left" - "ya" jika karyawan akhirnya keluar, "tidak" jika tidak.

3.2 Skeario penelitian

Penelitian ini bertujuan untuk memberikan wawasan dalam melakukan Penerapan algoritma SVM dalam memprediksis dan memperoleh model yang dapat memberikan hasil yang dapat diandalkan dan konsisten dalam kasus ini. Pada penelitian ini terdapat beberapa tahapan yang dimulai dari pembersihan data, ekstraksi fitur, serta penggunaan *metode balancing* untuk menangani ketidakseimbangan dalam distribusi kelas. Selain itu, eksperimen ini juga menggunkan metode pemisahan data dengan K-Fold untuk memastikan evaluasi yang komprehensif atau meyeluruh.

Berikut penjelasan mengenai skenario penelitian:



Gambar 2. 3 Skenario penelitian

1. Masukkan Dataset Langkah pertama yang dilakukan pada penelitian ini adalah memasukkan dataset yang akan digunakan untuk penelitian. Dataset ini dapat berisi informasi mengenai *employee turnover*.
2. pada tahap processing peneliti melakukan berbakai tahap seperti Cleaning (mengecek duplikat,null, Transformation dan Normalisasi).preprocessing dilakukan untuk membersihkan data dari nilai yang hilang atau tidak valid (cleaning), mentransformasi variabel jika diperlukan (transformation), normalisasi data jika skala variabel berbeda (normalisasi).
3. Balancing (SMOTE) Data balancing dilakukan menggunakan metode Synthetic Minority Over-sampling Technique (SMOTE). SMOTE menciptakan sampel sintetis dari kelas minoritas untuk menyeimbangkan distribusi kelas.
4. *Feature Extraction* digunakan untuk mengurangi dimensi dari dataset.pada *Feature Extraction* terdapat berbagai medote dan pada penelitian ini menggunakan metode Principal Component Analysis (PCA).
5. *Feature Selection* digunkan untuk merangkin setiap fitur berdasarkan relansinya terhadap *employee turnover* dengan menngunkan metode RFE.*Feature selection* juga digunakan untuk memeilih jumlah fitur yang akan digunakan dalam tahap pemodelan.
6. Pada tahap Permodelan dilakukan menggunakan algoritma klasifikasi *Support Vector Machine*. Model ini digunakan untuk mempelajari pola dari data training dan kemudian melakukan prediksi pada data test.
7. Pengukuran Hasil menggunakan Confusion Matrix Evaluasi model dilakukan dengan mengukur kinerja menggunakan Confusion Matrix.

Confusion Matrix memberikan informasi tentang *True Positive, True Negative, False Positive, dan False Negative*. Dari sini, berbagai metrik evaluasi seperti akurasi, presisi, recall, F1-score, dan area di bawah kurva ROC (ROC AUC) dapat dihitung.

Dengan mengikuti skenario ini, penelitian dapat memberikan pemahaman yang lebih baik tentang efektivitas model SVM dalam memprediksi churn customer, dengan mempertimbangkan berbagai teknik preprocessing, balancing data,ekstraksi fitur, dan evaluasi model yang lengkap.

3.2.1 Analisis Data

Pada tahap ini, dilakukan eksplorasi data untuk memahami karakteristik employee turnover. Variabel target yang akan diprediksi adalah employee turnover (No untuk tetap, Yes untuk berpindah). data ini terdiri 10 kolom dan 9540 baris, dengan kelas No berjumlah 6756 dan 2784 di kelas Yes.dilihat dari ketimpangan data antara kelas No dan Yes, maka deperlukan tahap balancing guna memepermudah penelitian.

3.2.2 Preprocessing

Proses preprocessing mencakup penanganan missing values, encoding variabel kategorikal, dan normalisasi data. Missing values dapat diisi dengan nilai rata-rata atau median. Variabel kategorikal diubah menjadi representasi numerik, dan data numerik dapat dinormalisasi untuk memastikan setiap variabel memiliki skala yang serupa.berikut bebrapa tahapan preprocessing yang dilakauakan.

1. data cleaning
    - penghapusan duplikat : Menghilangkan baris data yang duplikat untuk mencegah pengaruh yang tidak diinginkan pada analisis
    - mengecek null : untuk menghitung jumlah nilai null atau missing values dalam setiap kolom dari DataFrame df, pada data set ini tidak terdapat null atau *missing value*

Tabel 3. 1Mengecek nilai null

Fitur	null
department	0
promoted	0
review	0
projects	0
salary	0
tenure	0
satisfaction	0
bonus	0
avg_hrs_month	0
left	0

2. data transformation
   - Label encoding : mengubah fitur kategorikal menjadi numerik menngunkan labal encoding (department,salary dan left)

Tabel 3. 2Data sebelemu Transformation

department	salary	left
operations	low	no
suporrt	medium	yes

logistics	high	yes
IT	medium	no

Tabel 3. 3Data sesudah Transformation

department	salary	left
6	0	0
9	1	1
4	2	1
0	1	0

- Pada tahap ini membersihkan data di dua kolom, yaitu 'avg_hrs_month' dan 'satisfaction'. Pertama, kita mengganti koma (,) dengan string kosong (tidak ada karakter) dan titik (.) dengan koma (,) di kedua kolom tersebut. Setelah itu, nilai-nilai di kedua kolom dikonversi menjadi string agar dapat dilakukan penggantian karakter. Kemudian, nilai-nilai tersebut dikonversi kembali ke tipe data float.

Tabel 3. 4Data sebelum Transformation

satisfaction	avg_hrs_month
0.6267589740293295	1.808.660.696.668.470
0.4436789547574034	1.827.081.489.616.220
0.4468232240377964	1.844.160.840.365.650

Tabel 3. 5Data setelah Transformation

satisfaction	avg_hrs_month
6.267590e+15	1.808661e+15
4.436790e+15	1.827081e+15
4.468232e+15	1.844161e+15

3. data normalization
   - Min-Max Scaling: melakukan normalisasi pada fitur-fitur tertentu dalam dataset menggunakan metode Min-Max Scaling. Proses normalisasi dilakukan untuk menjadikan nilai-nilai fitur- fitur tersebut memiliki rentang antara 0 dan 1. Setiap fitur yang dipilih akan dinormalisasi secara independen. Hasil normalisasi tersebut kemudian digunakan untuk menggantikan nilai-nilai asli fitur-fitur dalam dataset.berikut contoh penerapanya:

Tabel 3. 6data sebelum normalization

satisfaction	avg_hrs_month
6.267590e+15	1.808661e+1
4.436790e+15	1.827081e+15

4.468232e+15	1.844161e+15

Tabel 3. 7 Data setelah di normalisasi

satisfaction	avg_hrs_month
0.626759	0.091070
0.443679	0.091998
0.446823	0.092858
0.440139	0.095019

3.2.3 Balancing

Data yang telah ditransformasi dilanjutkan dengan balancing karena kelas pada dataset yang digunakan tidak seimbang, proses balancing menggunakan metode SMOTE (Synthetic Minority Over-sampling Technique). Balancing dataset penting terutama jika terdapat ketidakseimbangan jumlah sampel antara kelas mayoritas dan kelas minoritas. SMOTE adalah metode yang menghasilkan sampel sintetis untuk kelas minoritas, sehingga meningkatkan keseimbangan dataset. Namun pada penelitian ini hasil yang diperoleh hasil mengalami penurunan dibandingkan dengan permodelan yang tidak melakukan balancing.

3.2.4 Feature Extraction

Pada tahap Feature Extraction, dengan menngunkan Principal Component Analysis (PCA) adalah teknik yang digunakan untuk mengurangi dimensi dataset. PCA merubah variabel-variabel awal menjadi kombinasi linear yang disebut principal components.

contoh penerapan fitur ekstraksi dengan PCA :

Tabel 3. 8data sebelum di ekstraksi

review	projects
0.387781	0.333333
0.640434	0.333333
0.597896	0.333333

Tabel 3. 9data setelah di ekstraksi

review	projects
-0.756740	1.013340
0.380807	-0.268620
0.437826	0.006824

3.2.5 *Feature Selection* dengan *Recursive feature Elimination* (RFE)

Pada tahap ini peneliti menggunakan RFE dalam seleksi fitur ,teknik ini digunakan untuk meranking setiap fitur dan memilih jumlah fitur yang akan di gunakan pada tahap pemodelan. Untuk meranking setiap fitur yang paling relevan Feature Selection Dengan RFE akan melakuakn beberapa tahapan, berikut adalah tahapanya:

Langkah-langkah Perhitungan RFE:

1. Inisialisasi Model:
   - Model dasar yang digunakan adalah *Support Vector Machine (SVM)* dengan kernel linear. Model ini dipilih karena kemampuannya memberikan bobot yang jelas untuk setiap fitur.
2. Proses Iteratif RFE:
   - RFE melatih model pada semua fitur yang tersedia, mengevaluasi pentingnya setiap fitur, dan menghapus fitur paling tidak penting satu per satu.
3. Pelatihan Model:
   - Model dilatih menggunakan semua fitur yang tersedia. SVM dengan kernel linear akan menyesuaikan bobot untuk setiap fitur berdasarkan kontribusinya terhadap prediksi.
4. Evaluasi Kepentingan Fitur:
   - Pentingnya setiap fitur dievaluasi berdasarkan koefisien bobot dari SVM. Fitur dengan bobot kecil dianggap kurang penting.
5. Penghapusan Fitur:
   - Fitur dengan bobot terendah dihapus dari dataset.
6. Pengulangan Proses:
   - Langkah-langkah di atas diulangi sampai semua fitur diurutkan atau jumlah fitur yang diinginkan tercapai.

Contoh Perhitungan RFE

1. Awal:
    - sembilan fitur: department_encoded, promoted, salary, review, satisfaction, bonus, avg_hrs_month, projects, dan tenure.
2. Iterasi Pertama:
    - Model SVM dilatih dengan kesembilan fitur tersebut.
    - Koefisien bobot untuk setiap fitur akan dievaluasi. Misalkan tenure memiliki koefisien bobot terkecil dan dihapus.
3. Iterasi Kedua:
    - Model dilatih ulang dengan delapan fitur yang tersisa.
    - Fitur dengan bobot terkecil berikutnya dihapus.
4. Iterasi Ketiga dan Seterusnya:
    - Proses berlanjut hingga hanya fitur yang paling penting yang tersisa.

Hasil Akhir
Peringkat fitur dari yang paling relevan hingga yang paling tidak relevan adalah:

1. department_encoded
2. promoted
3. salary
4. review
5. satisfaction
6. bonus
7. avg_hrs_month
8. projects
9. tenure

- Fitur Teratas yang Dipilih

Dari hasil peringkat tersebut, tujuh fitur teratas yang dipilih adalah:

1. department_encoded
2. promoted
3. salary
4. review
5. satisfaction
6. bonus
7. avg_hrs_month

3.2.6 Penyetelan Parameter

Penyetelan parameter modeling melibatkan pencarian kombinasi parameter terbaik yang meningkatkan kinerja model SVM. tuning parameter dapat dilakukan dengan eksplorasi nilai yang berbeda untuk jenis kernel(linear,rbf,sigmoid), C(0.1,1.0,10), dan gamma (0.1,1.0,10). Proses ini memastikan parameter yang paling optimal, sebelumnya peneliti ingin melihat berapakah rata rata akurasi tertinggi yang bisa peneliti dapatkan dengan mengkombinasikan hyperparameter, jumlah fitur dan jumlah k. jika terdapat hasil yang memiliki hasil yang sama maka peneliti akan memilih kombinasi yang lebih cepat dalam menjalankan program kombinasinya. Jika kombinasi dengan rata rata akurasi tertinggi terbilang sangat lama untuk menjalankan programnya, maka peneliti akan memilih rata rata akurasi tertinggi di setelah kombinasi tersebut.

3.2.7 Modeling

Pada tahap ini, setelah mem-preprocesing data, ekstraksi fitur, dan menyetel parameter, data data akan diklasifikasi dan peneliti akan mencoba berbagai kombinasi antara pameter,jumlah fitur dan kfold.untuk lebih memahami cara kerja algoritma SVM, peneliti mencoba melakuakan perhitungan manual dari algoritma SVM,peneliti menggunkan algoritma SVM linear sebagai contoh perhitungan manualnya.Pada tabel

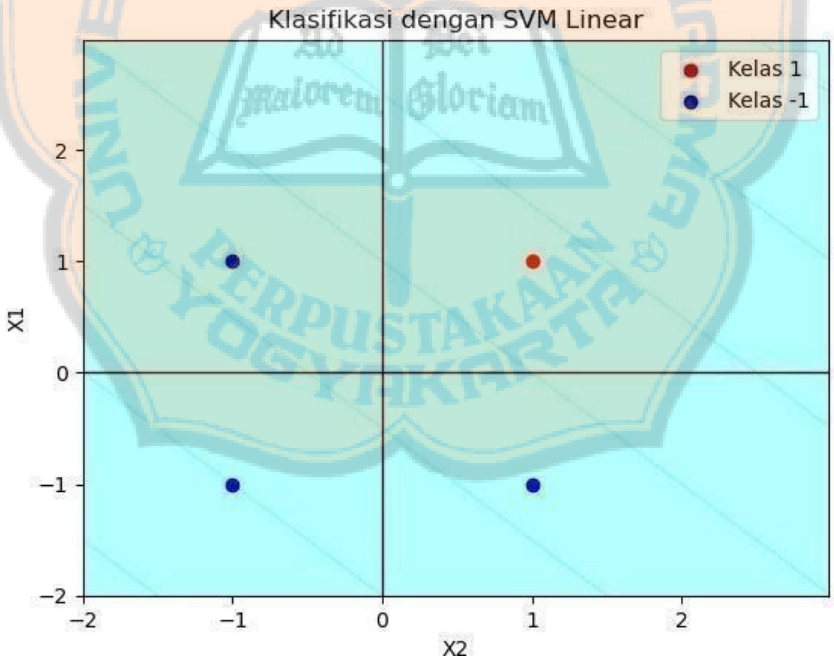
3.1 merupakan contoh data yang akan digunakan dalam perhitungan manual.

Tabel 3. 10 Data Yang Digunakan Untuk Perhitungan SVM Linear

x_1	x_2	Kelas (y)
1	1	1
-1	1	-1
1	-1	-1
-1	-1	-1

Pada tabel 3.10 ditampilkan data yang memiliki 2 fitur yaitu fitur $x_1$dan x_2 dan 2 kelas yaitu kelas 1 dan kelas -1. Pada SVM linear, jumlah fitur memengaruhi banyaknya bobot, dengan demikian akan ada bobot dengan jumlah yang sama yaitu 2 (w_1 dan w_2).

Visualisasi untuk data yang digunakan dalam perhitungan SVM linear ini akan digambarkan pada gambar 3.2



Gambar 3. 1 Visualisasi Data

Berdasarkan data pada tabel 3.10 dan gambar yang divisualisasikan pada gambar 3.1 nilai margin akan diminimalkan dengan persamaan (3.1)

$$\frac{1}{2}||w||^2 = \frac{1}{2}(w_1^2 + w_2^2) \qquad (3.1)$$

Dengan syarat:

$$y_i\,(w.x_i + b) \geq 1, \quad i = 1,2,3,\ldots,n$$

Berdasarkan jumlah fitur dan bobot yang dimiliki, dari pernyataan tersebut maka didapatkan:

$$y_i\,(w.x_1 + w.x_2 + b) \geq 1 \qquad (3.2)$$

Maka dapat dibuatkan persamaan:

$(w_1 + w_2 + b) \geq 1$ Untuk $x_1 = 1, x_2 = 1, y_1 = 1$ (3.3)

$(w_1 - w_2 - b) \geq 1$ Untuk $x_1 = -1, x_2 = 1, y_1 = -1$ (3.4)

$(-w_1 + w_2 - b) \geq 1$ Untuk $x_1 = 1, x_2 = -1, y_1 = -1$ (3.5)

$(-w_1 - w_2 - b) \geq 1$ Untuk $x_1 = -1, x_2 = -1, y_1 = -1$ (3.6)

Dari persamaan 3.3 hingga 3.6, mencari nilai bobot dan bias menggunakan persamaan tersebut. Disini akan digunakan teknik penjumlahan untuk mengeliminasi bobot ataupun bias agar menemukan bobot atau bias yang ingin ditemukan.

1. Mencari w_1 dengan cara menjumlahkan persamaan 3.3 dan 3.4

$$\begin{array}{l} (w_1 + w_2 + b) \geq 1 \\ \underline{(w_1 - w_2 - b) \geq 1} \; + \\ 2w_1 = 2 \\ w_1 = 1 \end{array}$$

2. Mencari w_2 dengan cara menjumlahkan persamaan 3.3 dan 3.5

$$(w_1 + w_2 + b) \geq 1$$

$$\underline{(-w_1 + w_2 - b) \geq 1} + \quad 2w_2 = 2$$

$$w_2 = 1$$

3. Mencari b dengan cara menjumlahkan persamaan 3.4 dan 3.5

$$(w_1 - w_2 - b) \geq 1$$
$$\underline{(-w_1 + w_2 - b) \geq 1} +$$
$$-2b = 2$$
$$b = -1$$

Setelah mendapatkan nilai w_1, w_2, dan b (bobot dan bias), berdasarkan persamaan 3.2 akan diperoleh persamaan *hyperplane* berikut:

$$w_1 . x_1 + w_2 . x_2 + b = 0$$

$$1 . x_1 + 1 . x_2 + (-1) = 0 \; x_1 + x_2 - 1 = 0$$

$$x_1 + x_2 = 1 \; x_2 = 1 - x_1$$

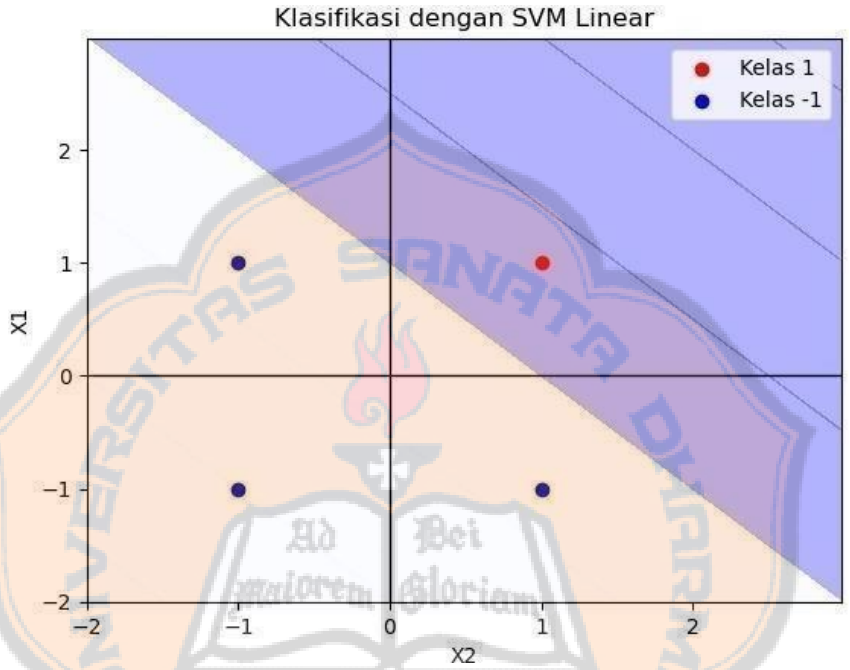
Sehingga untuk membentuk *hyperplane* berdasarkan nilai x_2 yang telah ditemukan dapat dilihat pada tabel 3.11.

Tabel 3. 11 Tabel Nilai Dari Hyperplane

x_1	x_2
-2	1-(-2) = 3
-1	1-(-1) = 2
x_1	x_2
0	1 - 0 = 1
1	1 - 1 = 0

2	1 - 2 = -1

Berdasarkan tabel 3.11, *hyperplane* yang terbentuk dari nilai bobot dan bias menjadi x_1 dan x_2, akan divisualisasikan pada gambar 3.3.



Gambar 3. 2Visualisasi Data Yang Telah Terpisahkan Hyperplane

Selanjutnya untuk memprediksi data yang ingin diprediksi, fungsi keputusan SVM linear dinyatakan $f(x) = SIGN\ (w.\ x + b)$. Berdasarkan nilai $w_1 = 1$, $w_2 = 1$, dan $b = -1$, maka fungsi keputusan linear akan dinyatakan $f(x) = x_1 + x_2 - 1$.

Sehingga dalam memprediksi data yang sama seperti yang digunakan untuk klasifikasi pada tabel 3.10, akan ditampilkan pada tabel 3.12.

Tabel 3. 12 Contoh Prediksi SVM.

x_1	x_2	Prediksi
1	1	$SIGN\ (x) = 1 + 1 - 1 = 1$
-1	1	$SIGN\ (x) = -1 + 1 - 1 = -1$
1	-1	$SIGN\ (x) = 1 + (-1) - 1 = -1$
-1	-1	$SIGN\ (x) = -1 + (-1) - 1 = -1$

3.2.6 K-fold Cross-Validation

Pada metode ini K-fold Cross-Validation membantu mengukur sejauh mana model dapat generalisasi pada data yang berbeda-beda. Dengan membagi data menjadi k lipatan, setiap iterasi melibatkan pemisahan satu lipatan sebagai data uji dan sisanya sebagai data pelatihan. Hal ini memastikan model diuji pada variasi yang lebih luas dari dataset, membantu mengidentifikasi apakah model cenderung overfit atau underfit pada data tertentu.

3.2.7 Confusion Matrix

pada metode ini memberikan pandangan yang lebih mendetail tentang performa model. Dengan menggunakan contoh tabel confusion matrix, kita dapat lebih dalam memahami bagaimana model SVM mengklasifikasikan karyawan sebagai berpindah atau tetap. Informasi ini digunakan untuk menghitung metrik evaluasi seperti akurasi, presisi, recall, dan F1-score, memberikan gambaran tentang kemampuan model dalam memprediksi employee turnover dengan tepat.

Tabel 3. 13 Confusion Matrix

confusion matriks		nilai sebenarnya	
		True	False
hasil prediksi	True	TP	TN
	False	FP	FN

Ket:

1. True Positive (TP): 5369 (Prediksi benar untuk kelas 1)
2. False Positive (FP): 677 (Prediksi salah untuk kelas 1, padahal kelas sebenarnya adalah 0.0)
3. False Negative (FN): 1387 (Prediksi salah untuk kelas 0, padahal kelas sebenarnya adalah 1)
4. True Negative (TN): 6079 (Prediksi benar untuk kelas 0)

Berdasarkan tabel 3.4 diperoleh hasil perhitungan *accuracy*, *precision*, *recall*, dan *f1-score*:

1. Akurasi

$$\text{Akurasi} = \frac{TP+TN}{TP+TN+FP+FN}$$

$$\text{Akurasi} = \frac{5.369+6.070}{5.369+6.079+677+1.387} = \frac{11.448}{13.612} \approx 0.84 = 84\%$$

2. Presisi untuk kelas 1

$$\text{presisi} = \frac{TP}{TP+FP}$$

$$\text{presisi} = \frac{5.369}{5.369+677} = \frac{5.369}{6.046} \approx 0.89 = 89\%$$

3. Recal untuk kelas 1

$$\text{Recal} = \frac{TP}{TP+FN}$$

$$\text{Recal} = \frac{5.369}{5.369+1.387} = \frac{5.369}{66.756} \approx 0.79 = 79\%$$

4. F1-Score untuk kelas 1

$$\text{F1-Score} = 2 \times \frac{Presisi \times \text{Recal}}{Presisi+Recal}$$

$$\text{F1-Score} = 2 \times \frac{0.89 \times 0.79}{0.89+0.79} \approx 2 \times \frac{0.7031}{1.68} \approx 0.84 = 84\%$$

3.3 Skenario pengujian

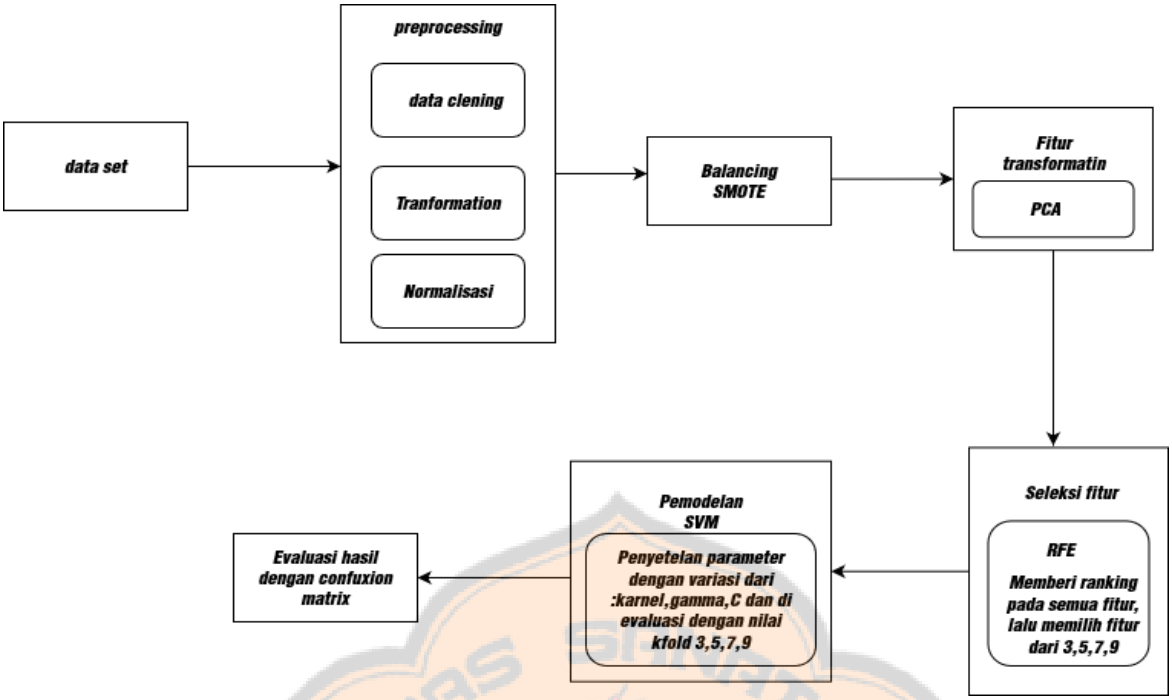
Pada penelitian ini, peneliti melakukan eksplorasi penggunaan algoritma Support Vector Machine (SVM) dengan variasi parameter untuk mencari konfigurasi terbaik yang memberikan hasil akurasi tinggi. Penelitian ini mencakup beberapa tahap, dimulai dari memanggil data set *employee turnover* ,kemudian kemudian data akan di preprocessing, proses ini diperlukan untuk membersihkan data yang nilainya hilang, duplikat, tidak valid, mengubah variabel, menormalkan data yang memiliki perbedaan skala.

Setelah peneliti telah memastikan data tidak sudah seelsai di *preprocessing*,selanjutnya peneliti melakukan *balancing* terhadap dataset

selanjutnya data akan ditransformasi menggunakan Principal Component Analysis (PCA). PCA digunakan untuk mengidentifikasi kombinasi liner yang memiliki variansi maksimum, setelah proses transformasi data selesai maka dilanjutkan dengan pemilihan atau seleksi fitur.

sebelum data set digunakan untuk tahap pemodelan ,peneliti melakukan seleksi fitur dengan mengunakan RFE.tahap ini dilakukan untuk meranking setiap fitur dan memilih fitur yang akan digunakan pada tahap pemodelan

Peneliti kemudian menjalankan model SVM dengan kombinasi antara parameter dari SVM yaitu variasi kernel, (linear, rbf, dan sigmoid),C(0.1, 1.0, 10) dan gama(0.1, 1.0, 10), kombinasi antara parameter SVM kemudian peneliti memasukan validasi silang (cross-validation) dengan menggunakan k-fold. Pada penelitian ini, Menggunakan beberapa k-fold 3, 5, 7 dan 9, yang merupakan nilai k-fold yang umum digunakan dalam praktik pengujian model. Setiap kombinasi dievaluasi dengan menghitung akurasi menggunakan mean_accuracy, dan hasilnya disimpan dalam dictionary. kombinasi antara parameter dengan akurasi tertinggi kemudian dipilih sebagai yang terbaik.



Gambar 3. 3 skenario pengujian

BAB IV

HASIL DAN ANALISIS PENELITIAN

Pada bab IV ini akan diberikan pemaparan mengenai hasil penelitian dan pembahasan yang telah dilakukan oleh peneliti saatprediksi menngunkan *svm* dalam memprediksi *employee turnover*. Penjelasan yang diberikan merupakan menjabaran dari rumusan masalah yang ada

4.1 Dataset dan *Preporcessing*

Langkah awal dalam implementasi adalah memasukan dataset, dengan menggunakan library pandas untuk membaca dataset dari file CSV yang teletak pada direktori D dengan path D:/skripsi upung/employee_churn_data.csv. Dataset tersebut dikumpulkan HRD dari tahun 2016-2020 dan berisi informasi tentang karyawan dengan jumlah 9.540 data . Data ini mencakup informasi Departemen yang menjadi tempat kerja karyawan, apakah mereka dipromosikan dalam 24 bulan terakhir, skor evaluasi terakhir yang diterima, jumlah proyek yang diikuti, tingkat gaji (rendah, sedang, tinggi), masa kerja dalam tahun, tingkat kepuasan karyawan dari survei, apakah mereka menerima bonus dalam 24 bulan terakhir, rata-rata jam kerja per bulan, dan apakah mereka akhirnya meninggalkan perusahaan, semuanya merupakan data yang dikumpulkan. Dataset diunduh dari situs Kaggle, dengan total 9.540 baris dan 10 kolom atribut. Keputusan menggunakan dataset *employee turnover* diambil karena ukuran dataset yang besar, dengan kombinasi atribut berupa tipe data integer dan kategori yang relevan dengan kebutuhan data mining dalam konteks bisnis. Berikut penerapan berupa source code yang digunakan untuk memanggil data tersebut.

:

	department	promoted	review	projects	salary	tenure	satisfaction	bonus	avg_hrs_month	left
0	operations	0	0.577569	3	low	5.0	0.6267589740293295	0	1.808.660.696.668.470	no
1	operations	0	0.751900	3	medium	6.0	0.4436789547574034	0	1.827.081.489.616.220	no
2	support	0	0.722548	3	medium	6.0	0.4468232240377964	0	1.844.160.840.365.650	no
3	logistics	0	0.675158	4	high	8.0	0.4401387461171622	0	1.887.075.447.757.310	no
4	sales	0	0.676203	3	high	5.0	0.5776074456916579	1	17.982.108.327.312.100	no
5	IT	0	0.683206	2	medium	5.0	0.5652518631559001	1	1.788.418.791.506.500	no
6	admin	0	0.620158	4	high	5.0	0.6869511701465508	0	18.114.295.615.536.400	no
7	support	0	0.499567	4	medium	7.0	0.720451118181815	1	18.497.753.752.486.000	no
8	sales	0	0.652818	4	low	6.0	0.6786959372102828	0	1.836.557.898.051.490	no
9	sales	0	0.642031	3	medium	6.0	0.6233650093739659	0	1.818.509.998.905.140	no

Gambar 4. 1 Source code pemanggilan data

Dari penerapan source code, akan dilanjutkan dengan menampilkan data berupa atribut dan isinya

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('darkgrid')
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.preprocessing import LabelEncoder

path_dataset= "D:/skripsi upung/employee_churn_data.csv"
df = pd.read_csv(path_dataset,sep =';')
df.head(10)
```

Gambar 4. 2Tampilan dataset

Hasil yang akan ditampilkan berupa informasi dari setiap atribut dan isinya, hasil yang ditampilkan bukan data secara keseluruhan hanya menampilkan 10 baris pertama pada dataset.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9540 entries, 0 to 9539
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   department      9540 non-null   object
 1   promoted        9540 non-null   int64
 2   review          9540 non-null   float64
 3   projects        9540 non-null   int64
 4   salary          9540 non-null   object
 5   tenure          9540 non-null   float64
 6   satisfaction    9540 non-null   object
 7   bonus           9540 non-null   int64
 8   avg_hrs_month   9540 non-null   object
 9   left            9540 non-null   object
dtypes: float64(2), int64(3), object(5)
```

Gambar 4. 3 info data set

Pada data "*Employee turnover*" memiliki baris 9.540 baris dan 10 kolom,memilki tipe data dari masing masing atribut int (3) , float (2) dan object (5).

4.1.1 Data Cleaning

Data cleaning adalah proses penting dalam analisis data di mana data yang tidak valid, tidak lengkap, tidak akurat, atau tidak relevan diidentifikasi dan diperbaiki atau dihapus dari kumpulan data. Tujuannya adalah untuk memastikan bahwa data yang digunakan untuk analisis atau pemodelan adalah berkualitas tinggi dan dapat diandalkan. Proses ini melibatkan tindakan seperti deteksi dan penanganan nilai yang hilang, validasi data,pembersihan format data, penghapusan duplikat.

```
df.duplicated().sum
```

```
<bound method NDFrame._add_numeric_operations.<locals>.sum of 0       False
1       False
2       False
3       False
4       False
       ...
9535    False
9536    False
9537    False
9538    False
9539    False
Length: 9540, dtype: bool>
```

Gambar 4. 4 Mengecek data Duplicate

pada gambar ini menampilkan bahwa tidak terdapat data *duplicated* pada data yang di gunakan.

```
] df.isnull().sum()

department        0
promoted          0
review            0
projects          0
salary            0
tenure            0
satisfaction      0
bonus             0
avg_hrs_month     0
left              0
dtype: int64
```

Gambar 4. 5 Mengecek Missing value

Pada gambar di atas menampilkan bahwa tidak terdapat *missing value* pada data yang digunakan.

4.1.2 Transformasi data

Data transformation adalah proses mengubah format atau struktur data agar lebih sesuai untuk analisis atau pemodelan. Hal ini sering diperlukan ketika algoritma tertentu tidak dapat mengolah data dalam format aslinya.dalam kasus ini Support Vector Machine (SVM) tidak cocok dengan dataset yang memiliki variabel string atau kategorikal. Oleh karena itu, sebagai bagian dari proses transformasi data, variabel-variabel tersebut perlu diubah menjadi format yang dapat diproses oleh SVM, misalnya dengan menggunakan teknik encoding

,normalisasi data dan mengubah tipe data. tujuan dari data transformation adalah untuk memastikan bahwa data yang digunakan dalam analisis memiliki kualitas yang baik dan sesuai dengan kebutuhan analisis yang dilakukan.

1. Mengubah data kategorikal menjadi numerik

Pada data ini ini ada beberapa atribut yang memiliki nilai data katergorikal, seperti department,Salary, dan left. Dikarenakan SVM (Support Vector Machine) adalah algoritma yang pada dasarnya bekerja dengan data numerik. Oleh karena itu, agar mempermudah penelitian ini

,Peneliti akan melakuakan perubahan terhadap beberapa atribut yang memiliki tipe data

	department	promoted	review	projects	salary	tenure	satisfaction	bonus	avg_hrs_month	left
0	operations	0	0.577569	3	low	5.0	0.6267589740293295	0	1.808.660.696.668.470	no
1	operations	0	0.751900	3	medium	6.0	0.4436789547574034	0	1.827.081.489.616.220	no
2	support	0	0.722548	3	medium	6.0	0.4468232240377964	0	1.844.160.840.365.650	no
3	logistics	0	0.675158	4	high	8.0	0.4401387461171622	0	1.887.075.447.757.310	no
4	sales	0	0.676203	3	high	5.0	0.5776074456916579	1	17.982.108.327.312.100	no
5	IT	0	0.683206	2	medium	5.0	0.5652518631559001	1	1.788.418.791.506.500	no
6	admin	0	0.620158	4	high	5.0	0.6869511701465508	0	18.114.295.615.536.400	no
7	support	0	0.499567	4	medium	7.0	0.720451118181815	1	18.497.753.752.486.000	no
8	sales	0	0.652818	4	low	6.0	0.6786959372102828	0	1.836.557.898.051.490	no
9	sales	0	0.642031	3	medium	6.0	0.6233650093739659	0	1.818.509.998.905.140	no

Gambar 4. 6 info dataset

Pada gambar di atas ada beberapa atribut yang memiliki tipe data Kategoorikal,selanjutnya peneliti akan mengganti atribut tersebut menjadi numerik.

```
# Kolom yang ingin diencode
categorical_columns = ['department']

# Menggunakan LabelEncoder untuk melakukan encoding
encoder = LabelEncoder()

# Melakukan encoding untuk setiap kolom kategorikal
for col in categorical_columns:
    df[col + '_encoded'] = encoder.fit_transform(df[col])

# Menghapus kolom yang belum diencode
df = df.drop(categorical_columns, axis=1)

# Menampilkan hasil encoding
print(df.head(10))
```

```
7]  # Label encodin
    df['salary'] = df.salary.replace({'low':0, 'medium':1,'high':2})
    df['left']   = df.left.replace({'yes':1,'no':0})
```

```
df.info
```

```
<bound method DataFrame.info of      promoted    review  projects  salary  tenure  satisfaction  bonus  \
0           0  0.577569         3       0     5.0      0.626758974029329       0
1           0  0.751900         3       1     6.0      0.443678954757403       0
2           0  0.722548         3       1     6.0      0.446823224037796       0
3           0  0.675158         4       2     8.0      0.440138746117162       0
4           0  0.676203         3       2     5.0      0.577607445691657       1
...       ...       ...       ...     ...     ...                   ...     ...
9535        0  0.610988         4       1     8.0      0.543640597947436       0
9536        0  0.746887         3       1     8.0      0.549048466989071       0
9537        0  0.557980         3       0     7.0      0.705424760532802       0
9538        0  0.584446         4       1     8.0      0.607286977749824       1
9539        0  0.626373         3       0     7.0      0.706454786038683       1

                avg_hrs_month  left  department_encoded
0      1.808.660.696.668.470     0                   6
1      1.827.081.489.616.220     0                   6
2      1.844.160.840.365.650     0                   9
3      1.887.075.447.757.310     0                   4
4     17.982.108.327.312.100     0                   8
...                      ...   ...                 ...
9535   1.881.557.382.137.330     1                   6
9536   1.881.761.641.686.730     1                   4
9537  18.653.100.819.063.800     1                   6
9538   1.876.413.697.680.350     1                   0
9539  18.592.093.422.911.900     1                   3

[9540 rows x 10 columns]>
```

Gambar 4. 7 mengbah data kategorikal menjadi numerik

Pada gambar ini peneliti menunjukan program untuk mengubah tipe data kategorikal,program ini menggunkan label encoding dalam mengubah tipe data. pada program ini Peneliti melakukan 2 kali metode label encode.Pada bagian pertama kode, digunakan encoding pada kolom 'department'. Encoder ini akan

memberikan nilai numerik unik untuk setiap kategori di kolom tersebut. Misalnya, jika 'HR' diwakili oleh nilai 0, 'IT' oleh nilai 1, dan seterusnya. pada bagian kedua kode, digunakan metode replace untuk mengonversi nilai kategori pada kolom 'salary' dan 'left' secara manual. Nilai 'low' pada kolom 'salary' diubah menjadi 0, 'medium' menjadi 1, dan 'high' menjadi 2. Pada kolom 'left', nilai 'yes' diubah menjadi 1 dan 'no' menjadi 0.

2. Mengubah tipe data

```
df.info
```

```
<bound method DataFrame.info of       promoted    review  projects  salary  tenure        satisfaction  bonus  \
0            0  0.577569         3       0     5.0  0.6267589740293295      0
1            0  0.751900         3       1     6.0  0.4436789547574034      0
2            0  0.722548         3       1     6.0  0.4468232240377964      0
3            0  0.675158         4       2     8.0  0.4401387461171622      0
4            0  0.676203         3       2     5.0  0.5776074456916579      1
...        ...       ...       ...     ...     ...                 ...    ...
9535         0  0.610988         4       1     8.0  0.5436405979474364      0
9536         0  0.746887         3       1     8.0   0.549048466989071      0
9537         0  0.557980         3       0     7.0  0.7054247605328025      0
9538         0  0.584446         4       1     8.0  0.6072869777498247      1
9539         0  0.626373         3       0     7.0   0.706454786038683      1

              avg_hrs_month  left  department_encoded
0      1.808.660.696.668.470     0                   6
1      1.827.081.489.616.220     0                   6
2      1.844.160.840.365.650     0                   9
3      1.887.075.447.757.310     0                   4
4     17.982.108.327.312.100     0                   8
...                      ...   ...                 ...
9535   1.881.557.382.137.330     1                   6
9536   1.881.761.641.686.730     1                   4
9537  18.653.100.819.063.800     1                   6
9538   1.876.413.697.680.350     1                   0
9539  18.592.093.422.911.900     1                   3
```

Gambar 4. 8 info dataset sebelum tipe data dirubah

Pada gambar di atas merupakan data info sebelum atribut avg_hrs_month dan satisfaction sebelum di rubah.

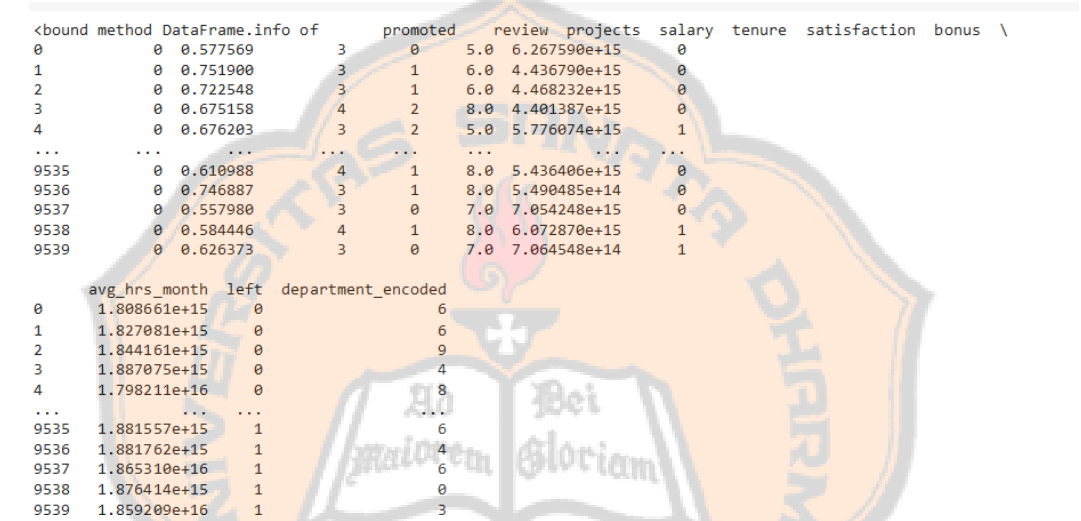
```
import pandas as pd

# Mengganti ',' dengan '' dan '.' dengan ',' pada kolom 'avg_hrs_month' dan 'satisfaction'
df[['avg_hrs_month', 'satisfaction']] = df[['avg_hrs_month', 'satisfaction']].astype(str).replace('[.,]', '', regex=True).astype(float)

# Filtering baris dengan nilai numerik yang valid
df = df[pd.to_numeric(df['avg_hrs_month'], errors='coerce').notnull() & pd.to_numeric(df['satisfaction'], errors='coerce').notnull()]
```

Gambar 4. 9 Mengubah tipe data avg_hrs_month dan satisfaction

Pada tahap ini, peneliti melakukan perubahan pada dua kolom, yaitu 'satisfaction' dan 'avg_hrs_month', dalam DataFrame df. Terlihat adanya masalah pada format nilai dalam kolom 'avg_hrs_month', di mana nilai- nilai tersebut memiliki karakter pemisah ribuan berupa titik ('.'). Karena format tersebut, Python menganggapnya sebagai string, bukan angka, yang dapat mengakibatkan masalah saat melakukan operasi numerik atau analisis data. Untuk mengatasi hal ini, peneliti memilih kedua kolom tersebut, 'avg_hrs_month' dan 'satisfaction', mengonversi nilai-nilai menjadi tipe data string, dan kemudian melakukan penggantian setiap kemunculan karakter koma (',') atau titik ('.') dengan string kosong (''). Selanjutnya, nilai-nilai dalam kedua kolom diubah kembali menjadi tipe data float untuk memastikan kevalidan data numerik. Pada langkah terakhir, dilakukan proses filtering pada DataFrame df, di mana hanya baris-baris yang memiliki nilai numerik yang valid pada kolom 'avg_hrs_month' dan 'satisfaction' yang disertakan. Dengan demikian, baris-baris yang memiliki nilai non-numerik diubah menjadi NaN dan selanjutnya dihapus dari DataFrame. Langkah-langkah ini dilakukan agar

data menjadi konsisten dan siap untuk digunakan dalam analisis lebih lanjut.

Pada gambar ini merupakan setelah pembersihan data, terlihat bahwa sebagian besar nilai dalam kolom *'satisfaction'* diekspresikan dalam format numerik yang benar. Namun, beberapa nilai tampaknya tidak sesuai dengan skala umum, seperti nilai-nilai yang sangat tinggi seperti 6.267590e+15 pada kolom *'satisfaction'*. Masalah yang muncul, seperti nilai yang sangat tinggi pada kolom *'satisfaction'*, menunjukkan adanya ketidaksesuaian dalam data yang memerlukan penanganan lebih lanjut.



```
<bound method DataFrame.info of       promoted    review  projects  salary  tenure  satisfaction  bonus  \
0            0  0.577569         3       0     5.0  6.267590e+15      0
1            0  0.751900         3       1     6.0  4.436790e+15      0
2            0  0.722548         3       1     6.0  4.468232e+15      0
3            0  0.675158         4       2     8.0  4.401387e+15      0
4            0  0.676203         3       2     5.0  5.776074e+15      1
...        ...       ...       ...     ...     ...           ...    ...
9535         0  0.610988         4       1     8.0  5.436406e+15      0
9536         0  0.746887         3       1     8.0  5.490485e+14      0
9537         0  0.557980         3       0     7.0  7.054248e+15      0
9538         0  0.584446         4       1     8.0  6.072870e+15      1
9539         0  0.626373         3       0     7.0  7.064548e+14      1

      avg_hrs_month  left  department_encoded
0      1.808661e+15     0                   6
1      1.827081e+15     0                   6
2      1.844161e+15     0                   9
3      1.887075e+15     0                   4
4      1.798211e+16     0                   8
...             ...   ...                 ...
9535   1.881557e+15     1                   6
9536   1.881762e+15     1                   4
9537   1.865310e+16     1                   6
9538   1.876414e+15     1                   0
9539   1.859209e+16     1                   3
```

Gambar 4. 10 info atribut avg_hrs_month dan satisfaction sesudah di rubah

Untuk memperbaikinya, peneliti malakukan pendekatan dengan menggunakan normalisasi data teknik yang akan dipakau adalah *MinMaxScaler*. Normalisasi dapat membantu dalam menyesuaikan rentang nilai menjadi lebih sesuai, sehingga meminimalkan dampak dari nilai ekstrem atau outliers seperti itu.

3. normalisasi data

```
from sklearn.preprocessing import MinMaxScaler

# Menentukan fitur yang akan dinormalisasi
features = ['promoted', 'review', 'projects', 'salary', 'tenure', 'satisfaction', 'bonus', 'avg_hrs_month', 'left', 'department_encoded']
# Membuat objek MinMaxScaler
scaler = MinMaxScaler()

# Melakukan normalisasi pada fitur-fitur yang dipilih
df[features] = scaler.fit_transform(df[features])

# Menampilkan dataset setelah normalisasi
print(df.head())
```

```
   promoted    review  projects  salary  tenure  satisfaction  bonus  \
0       0.0  0.387781  0.333333     0.0     0.3      0.626759    0.0
1       0.0  0.640434  0.333333     0.5     0.4      0.443679    0.0
2       0.0  0.597896  0.333333     0.5     0.4      0.446823    0.0
3       0.0  0.529215  0.666667     1.0     0.6      0.440139    0.0
4       0.0  0.530729  0.333333     1.0     0.3      0.577607    1.0

   avg_hrs_month  left  department_encoded
0       0.091070   0.0            0.666667
1       0.091998   0.0            0.666667
2       0.092858   0.0            1.000000
3       0.095019   0.0            0.444444
4       0.905524   0.0            0.888889
```

Gambar 4. 11 setelah melakukan minmax scaller

Pada tahap ini, peneliti menggunakan teknik normalisasi data dengan *MinMaxScaler* pada setiap *fitur* . Normalisasi dilakukan untuk membawa nilai- nilai pada fitur tersebut ke dalam rentang antara 0 dan 1, sehingga mempermudah perbandingan dan interpretasi. Objek *MinMaxScaler* dibuat, dan fitur-fitur yang dipilih diubah menggunakan metode *fit_transform() dari scaler*. Hasil normalisasi ini kemudian ditampilkan pada dataset, di mana nilai setiap *fitur* telah diubah sedemikian rupa sehingga sesuai dengan skala yang diinginkan. Proses normalisasi ini dapat meningkatkan konsistensi dan performa model yang akan dikembangkan, terutama jika terdapat perbedaan skala yang signifikan di antarafitur-fitur tersebut sebelum normalisasi.

4.2 Balancing data

Pada tahap ini, peneliti menggunakan metode *SMOTE (Synthetic Minority Over- sampling Technique)* untuk menangani ketidakseimbangan kelas pada dataset. Dataset awal memiliki ketidakseimbangan dalam kelas 'left', dengan jumlah data pada kelas 1 (*left*) sebanyak 2784 dan kelas 0 (tidak left) sebanyak 6756. Menggunakan *SMOTE*. penggunaan *SMOTE* pada penelitian ini untuk mengatasi ketidakseimbangan kelas

yang dapat menyebabkan model cenderung bermasalah dalam mengidentifikasi kelas minoritas, sehingga dapat mengurangi performa dan keakuratan model. Dengan menyeimbangkan kelas menggunakan *SMOTE*, model yang dihasilkan memiliki kemampuan yang lebih baik dalam mengklasifikasikan kelas minoritas dan menghasilkan hasil yang lebih seimbang dan akurat secara keseluruhan.

```
# Menghitung jumlah data pada kelas 1 dan 0 dalam kolom 'left'
left_counts = df['left'].value_counts()

# Menampilkan hasil
print("Jumlah data pada kelas 1 (left):", left_counts[1])
print("Jumlah data pada kelas 0 (tidak left):", left_counts[0])
```

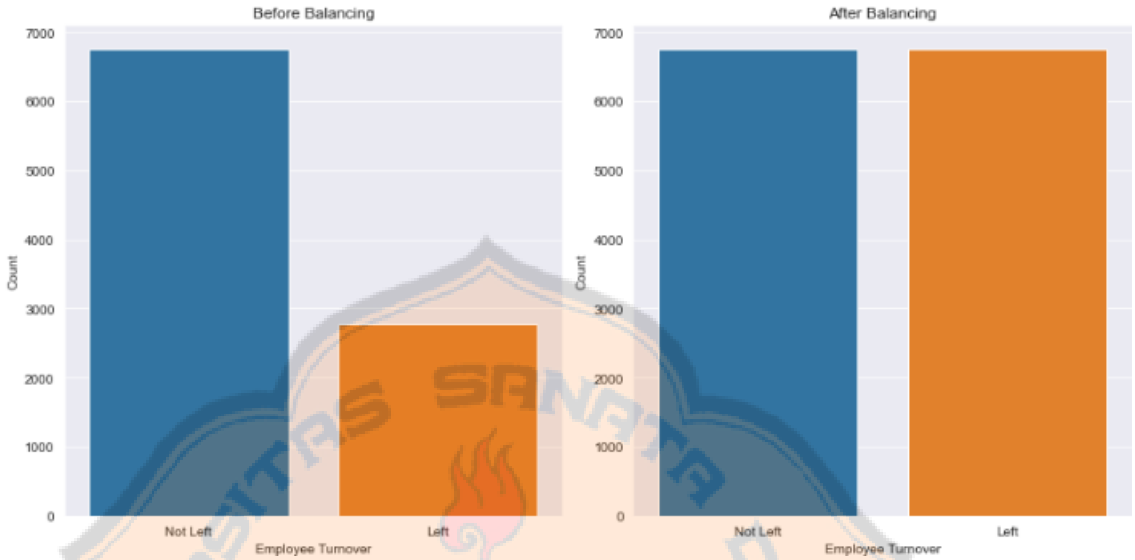
```
Jumlah data pada kelas 1 (left): 2784
Jumlah data pada kelas 0 (tidak left): 6756
```

Gambar 4. 12 Jumalah kelas left{1=left,0=tidak left)

```
In [18]: from imblearn.over_sampling import SMOTE
         import matplotlib.pyplot as plt
         import seaborn as sns

         # Inisialisasi SMOTE dengan random_state=42
         smote = SMOTE(random_state=42)
         X = df.drop('left', axis=1)
         y = df['left']
         # Oversampling menggunakan SMOTE
         X_resampled, y_resampled = smote.fit_resample(X, y)

         # DataFrame baru dari X_resampled
         df_resampled = pd.DataFrame(X_resampled, columns=X.columns)
         df_resampled['left'] = y_resampled
         # Hitung jumlah kelas pada dataset yang sudah di-balancing
         class_counts = df_resampled['left'].value_counts()

         # Visualisasi jumlah kelas setelah balancing
         plt.figure(figsize=(8, 6))
         sns.countplot(data=df_resampled, x='left')
         plt.xlabel('Employee Turnover')
         plt.ylabel('Count')
         plt.title('Balanced Employee Turnover Classes')
         plt.show()
         # Menghitung jumlah kelas sebelum dan sesudah balancing
         class_counts_before = y.value_counts()
         class_counts_after = df_resampled['left'].value_counts()

         # Visualisasi jumlah kelas sebelum dan sesudah balancing
         plt.figure(figsize=(12, 6))
         plt.subplot(1, 2, 1)
         sns.barplot(x=class_counts_before.index, y=class_counts_before.values)
         plt.xlabel('Employee Turnover')
         plt.ylabel('Count')
         plt.title('Before Balancing')
         plt.xticks([0, 1], ['Not Left', 'Left'])

         plt.subplot(1, 2, 2)
         sns.barplot(x=class_counts_after.index, y=class_counts_after.values)
         plt.xlabel('Employee Turnover')
         plt.ylabel('Count')
         plt.title('After Balancing')
         plt.xticks([0, 1], ['Not Left', 'Left'])
         # Menampilkan jumlah angka data sebelum balancing
         print("Jumlah data sebelum balancing:")
         print(class_counts_before)

         # Menampilkan jumlah angka data setelah balancing
         print("\nJumlah data setelah balancing:")
         print(class_counts_after)
         plt.tight_layout()
         plt.show()
```

Gambar 4. 13 Source code Balancing data

```
Jumlah data sebelum balancing:
0.0    6756
1.0    2784
Name: left, dtype: int64

Jumlah data setelah balancing:
0.0    6756
1.0    6756
Name: left, dtype: int64
```



Gambar 4. 14 output setelah dan sebelum balancing data

Setelah melakukan balancing dengan *SMOTE*, jumlah data antara kelas mayoritas dan kelas minoritas menjadi seimbang. Hal ini membantu meningkatkan kualitas prediksi model dengan mengurangi bias, serta memungkinkan model untuk mempelajari pola-pola yang relevan dari seluruh dataset. Sebelum proses balancing, terdapat 6756 data untuk kelas 0 dan 2784 data untuk kelas 1. Setelah balancing, kedua kelas memiliki jumlah data yang sama, yaitu 6756.

4.3 Feature Extraction

Pada langkah ini, peneliti melakukan *Feature extraction* dengan menggunakan Analisis Komponen Utama (PCA) untuk menyederhanakan dan menjelajahi dataset. PCA adalah metode statistik yang membantu mengidentifikasi pola utama dalam data, dengan tujuan mengurangi kompleksitasnya sambil mempertahankan sebagian besar informasi penting. Dengan memilih fitur-fitur tertentu dari dataset, PCA diterapkan untuk menciptakan representasi data yang lebih sederhana tanpa mengorbankan makna utama. Output utama yang dieksplorasi melibatkan varians ratio dan hasil *feature extraction*, memberikan wawasan lebih lanjut tentang struktur dataset yang digunakan.

```python
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

np.random.seed(42)

scaler = StandardScaler()

def feature_selection(df):
    X = df.drop('left', axis=1)
    y = df['left']

    # Standarisasi fitur
    scaler.fit(X)
    x_scaled = scaler.fit_transform(X)

    # Menggunakan semua fitur dalam PCA
    pca = PCA(n_components=X.shape[1], random_state=42)
    x_new = pca.fit_transform(x_scaled)
    columns = [f"PC{i+1}" for i in range(X.shape[1])]

    # Simpan nama-nama fitur sebelumnya
    feature_names = X.columns

    # Ganti nama hasil PCA dengan nama asli dari fitur
    df_new = pd.DataFrame(x_new, columns=columns)
    df_new = pd.concat([df_new, df['left']], axis=1)  # Menggabungkan hasil PCA dengan kolom 'left'

    # Tambahkan nama fitur asli ke DataFrame baru
    df_new.rename(columns=dict(zip(columns, feature_names)), inplace=True)

    return df_new

# Memanggil fungsi feature_selection untuk mendapatkan DataFrame yang sudah diekstraksi fiturnya menggunakan PCA
df_extracted = feature_selection(df_resampled)

# Pisahkan data menjadi atribut dan label
X_pca = df_extracted.drop('left', axis=1)
y = df_extracted['left']

# Tampilkan informasi dataset
print("Jumlah baris awal:", df_resampled.shape[0])
print("Jumlah baris setelah FE:", df_extracted.shape[0])
print("Data sampel setelah FE:")
print(df_extracted.head())
```

Gambar 4. 15 source code feature extraction

Peneliti mengimplementasikan analisis komponen utama (PCA) menggunakan pustaka scikit-learn untuk mengurangi dimensi data dan mengidentifikasi pola utama dalam dataset. Pertama, fungsi 'feature_selection' dipanggil dengan argumen DataFrame 'df_resampled'. Di dalam fungsi ini, fitur-fitur independen dipisahkan dari kolom target ('left'). Kemudian, fitur-fitur independen dinormalisasi dan disesuaikan menggunakan PCA tanpa melalui tahap standarisasi secara eksplisit. PCA digunakan dengan mengatur jumlah komponen utama menjadi sama dengan jumlah fitur asli dalam dataset. Hasil PCA, yang merupakan representasi data setelah reduksi dimensi, disimpan dalam sebuah DataFrame baru ('df_new'). Nama kolom pada DataFrame baru ini diganti dengan nama fitur asli. Setelah itu, DataFrame baru ini digunakan untuk memisahkan data menjadi atribut ('X_pca') dan label ('y'). Terakhir, informasi tentang dataset dicetak, termasuk jumlah baris sebelum dan sesudah proses ekstraksi fitur, serta lima baris pertama dari DataFrame yang sudah diekstraksi fiturnya. Ini memberikan gambaran tentang efektivitas ekstraksi fitur menggunakan PCA dalam mengelola dimensi data.

```
Jumlah baris awal: 13512
Jumlah baris setelah FE: 13512
Data sampel setelah FE:
   promoted    review  projects    salary    tenure  satisfaction     bonus  \
0 -0.687697 -0.756740  1.013340 -0.999522 -0.382428      0.029886  1.520169
1  0.819701  0.380807 -0.268620 -0.817632 -0.030452     -0.042923  0.328044
2  0.567910  0.437826  0.006824 -1.346071  0.041193      0.875176 -0.016904
3 -0.521237  1.246423 -0.754075  0.627082  0.863205     -0.875237 -1.534578
4  0.603740 -1.239998 -1.028117  0.639154 -1.499033      1.617982 -1.587667

   avg_hrs_month  department_encoded  left
0      -1.261860            0.915113   0.0
1      -0.943118           -0.405271   0.0
2      -0.924075           -0.187716   0.0
3      -0.735660           -0.538038   0.0
4       0.135146            0.058939   0.0
```

Gambar 4. 16 ouput setelah feature extraction

Perubahan yang dapat diamati melibatkan transformasi nilai-nilai fitur menjadi nilai-nilai baru dalam ruang dimensi yang lebih rendah setelah penerapan PCA.

4.4 Feature Selection

```python
from sklearn.feature_selection import RFE
from sklearn.svm import SVC

def feature_ranking_svm(X, y):
    # Inisialisasi model yang akan digunakan untuk RFE
    model = SVC(kernel='linear')  # Menggunakan kernel linear untuk SVM

    # Inisialisasi objek RFE
    rfe = RFE(model, n_features_to_select=1)  # Memilih satu fitur terbaik pada setiap iterasi

    # Melakukan seleksi fitur
    rfe.fit(X, y)

    # Mendapatkan peringkat fitur
    feature_ranking = rfe.ranking_

    # Mengurutkan fitur berdasarkan peringkat
    ranked_features_indices = feature_ranking.argsort()

    # Mendapatkan nama fitur yang terurut berdasarkan peringkat
    ranked_features = X.columns[ranked_features_indices]

    return ranked_features

# Memanggil fungsi untuk mendapatkan peringkat fitur menggunakan SVM
ranked_features_svm = feature_ranking_svm(X_pca, y)

# Menampilkan hasil peringkat fitur
print("Peringkat fitur dari yang paling relevan (SVM):")
for i, feature in enumerate(ranked_features_svm):
    print(f"{i+1}. {feature}")

# Pilih jumlah fitur teratas yang ingin digunakan
jumlah_fitur_teratas = 5

# Memilih jumlah fitur teratas dari hasil peringkat
fitur_teratas = ranked_features_svm[:jumlah_fitur_teratas]

# Menampilkan fitur teratas yang dipilih
print(f"Fitur teratas yang dipilih:")
for i, feature in enumerate(fitur_teratas):
    print(f"{i+1}. {feature}")

# Sekarang, gunakan X_pca_selected dalam pemodelan SVM
X_pca_selected = X_pca[fitur_teratas]  # Memilih hanya fitur teratas yang dipilih
```

Gambar 4. 17 source code feature selection

Pada tahap ini peneliti mencoba untuk mencari jumlah feature yang terpaling optimal. Semua variasi fitur tersebut terpilih menggunakan metode RFE dalam feature selection yang akan digunakan untuk mencari hasil akurasi tertinggi ataupun kombinasi *hyperparameter* yang terbaik dengan mengkombinasikan dengan hyperparameter C, gamma, Kernel dan akan di uji lebih lanjut dengan menggunakan metode *k-fold cross validation* dengan variasi k adalah 3, 5, 7 dan 9. proses seleksi fitur dilakukan dengan memanggil metode fit dari objek RFE (rfe.fit(X, y)), di mana X merupakan dataset fitur dan y adalah label kelas. Pada tahap ini, model SVM dengan kernel linear digunakan untuk menentukan relevansi setiap fitur terhadap label kelas. Selanjutnya, setelah proses seleksi selesai, peringkat fitur diperoleh dengan menggunakan atribut ranking_

dari objek RFE (feature_ranking = rfe.ranking_). Fitur-fitur kemudian diurutkan berdasarkan peringkatnya untuk mengetahui fitur mana yang paling relevan. Setelah itu, kita memilih jumlah fitur teratas yang ingin digunakan dan menampilkannya sebagai fitur teratas yang dipilih. Proses pemilihan fitur teratas dilakukan dengan cara mengambil sejumlah fitur teratas dari hasil peringkat fitur yang sudah diurutkan.pada line "jumlah_fitur_teratas = 5" peneliti akan 5 fitur tertas yang kana di gunakan pada tahap pemodelan, pada line ini bisa diganti ganti sesuai keingan peneliti, pada penelitian ini jumlah fitur yang akan di bandingkan adalan 3,5,7 dan 9.

```
Peringkat fitur dari yang paling relevan (SVM):
1. department_encoded
2. promoted
3. salary
4. review
5. satisfaction
6. bonus
7. avg_hrs_month
8. projects
9. tenure
Fitur teratas yang dipilih:
1. department_encoded
2. promoted
3. salary
```

Gambar 4. 18 output feature selection dengan 3 fitur

```
Peringkat fitur dari yang paling relevan (SVM):
1. department_encoded
2. promoted
3. salary
4. review
5. satisfaction
6. bonus
7. avg_hrs_month
8. projects
9. tenure
Fitur teratas yang dipilih:
1. department_encoded
2. promoted
3. salary
4. review
5. satisfaction
```

Gambar 4. 19 output selectio dengan 5 fitur

```
Peringkat fitur dari yang paling relevan (SVM):
1. department_encoded
2. promoted
3. salary
4. review
5. satisfaction
6. bonus
7. avg_hrs_month
8. projects
9. tenure
Fitur teratas yang dipilih:
1. department_encoded
2. promoted
3. salary
4. review
5. satisfaction
6. bonus
7. avg_hrs_month
```

Gambar 4. 20 ouput feature selection denga 7 fitur

```
Peringkat fitur dari yang paling relevan (SVM):
1. department_encoded
2. promoted
3. salary
4. review
5. satisfaction
6. bonus
7. avg_hrs_month
8. projects
9. tenure
Fitur teratas yang dipilih:
1. department_encoded
2. promoted
3. salary
4. review
5. satisfaction
6. bonus
7. avg_hrs_month
8. projects
9. tenure
```

Gambar 4. 21 output feature selection denga 9 fitur

4.5 Pemodelan SVM

peneliti mencoba mencari parameter terbaik untuk model SVM. Parameter- parameter yang dieksplorasi adalah C dan gamma, serta jenis kernel yang digunakan. Rentang nilai yang dicoba untuk C adalah 0.1, 1, dan 10, sedangkan untuk gamma adalah 0.01, 0.1, 1, dan 10. Tiga jenis kernel yang dipertimbangkan adalah 'linear', 'rbf', dan 'sigmoid'.

Selain itu, penelitian ini juga mempertimbangkan penggunaan berbagai jumlah fitur, yaitu 3, 5, 7, dan 9. Fitur-fitur ini dipilih dari hasil peringkat fitur yang diperoleh sebelumnya menggunakan metode seleksi fitur dengan *SVM-RFE*. Setiap kombinasi parameter dan jumlah fitur akan dievaluasi dengan menggunakan nilai k-fold cross- validation sebanyak 3, 5, 7, dan 9.

```
from sklearn.model_selection import KFold
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Definisikan daftar parameter yang akan diuji
kernel_values = ['linear','rbf','sigmoid']
C_values = [0.1, 1.0, 10]
gamma_values = [0.1, 1.0, 10]
k_values = [3,5,7,9]

results = []  # Menyimpan semua hasil kombinasi

for kernel_type in kernel_values:
    for C_value in C_values:
        for gamma_value in gamma_values:
            for k_fold in k_values:
                kf = KFold(n_splits=k_fold, shuffle=True,random_state=42)
                svm = SVC(kernel=kernel_type, C=C_value, gamma=gamma_value)
                scores = []

                for fold, (train_index, test_index) in enumerate(kf.split(X_pca_selected, y), 1):
                    X_train, X_test = X_pca_selected.iloc[train_index], X_pca_selected.iloc[test_index]
                    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

                    svm.fit(X_train, y_train)
                    y_pred = svm.predict(X_test)

                    score = accuracy_score(y_test, y_pred)
                    scores.append(score)

                mean_accuracy = sum(scores) / len(scores)
                results.append({
                    'kernel': kernel_type,
                    'C': C_value,
                    'gamma': gamma_value,
                    'k_fold': k_fold,
                    'accuracy': mean_accuracy
                })

# Mencetak semua hasil
for result in results:
    print(f"Kernel: {result['kernel']}, C: {result['C']}, Gamma: {result['gamma']},
          K-Fold: {result['k_fold']}, Accuracy: {result['accuracy']}")

# Mencetak parameter dengan akurasi terbaik
best_params = max(results, key=lambda x: x['accuracy'])
print("\nBest Parameters:")
print(f"Kernel: {best_params['kernel']}, C: {best_params['C']}, Gamma: {best_params['gamma']},
      K-Fold: {best_params['k_fold']}, Accuracy: {best_params['accuracy']}")
```

Gambar 4. 22 Pemodelan SVM

4.6 Confusion matrix

Setelah Peneliti melakukan analisis pada tahap modeling, langkah selanjutnya adalah Peneliti mengevaluasi performa model menggunakan *Classification Report* dan *Confusion Matrix*. Dalam melakukan analisis dan perbandingan hasil dari setiap fitur, kami mengamati performa model SVM pada empat kombinasi fitur yang berbeda: menggunakan 3, 5, 7, dan 9 fitur. Hasilnya menunjukkan bahwa, meskipun terdapat variasi dalam jumlah fitur yang digunakan, performa model relatif konsisten di seluruh kombinasi fitur.

Dengan menggunakan hanya 3 fitur, model mencapai akurasi sebesar 84%, dengan presisi untuk kelas 0 sebesar 81% dan kelas 1 sebesar 90%. Meskipun akurasi meningkat menjadi 85% ketika menggunakan 5, 7, atau 9 fitur, pola presisi dan recall tetap relatif stabil di seluruh kombinasi fitur tersebut. Hal ini menunjukkan bahwa penambahan fitur tidak secara signifikan mempengaruhi performa model secara keseluruhan.

Namun demikian, kami perhatikan bahwa ada sedikit variasi dalam jumlah prediksi benar untuk setiap kelas antara berbagai kombinasi fitur. Misalnya, terdapat peningkatan kecil dalam jumlah prediksi benar untuk kelas 1 ketika menggunakan 5 fitur, tetapi jumlah prediksi benar untuk kelas 0 sedikit menurun dibandingkan dengan menggunakan 3 fitur. Meskipun demikian, perbedaan ini tidak signifikan secara substansial.

Dengan demikian, hasil analisis menunjukkan bahwa penambahan fitur tidak secara krusial meningkatkan performa model SVM dalam kasus ini.

```
from sklearn.metrics import classification_report, confusion_matrix
# Menggunakan parameter terbaik yang telah ditemukan
best_kernel = best_params['kernel']
best_C = best_params['C']
best_gamma = best_params['gamma']
best_k_fold = best_params['k_fold']

# Membuat model SVM dengan parameter terbaik
best_svm_model = SVC(kernel=best_kernel, C=best_C, gamma=best_gamma, random_state=42)

# Membuat objek KFold untuk evaluasi
kf = KFold(n_splits=best_k_fold, shuffle=True, random_state=42)

# Inisialisasi list untuk menyimpan hasil prediksi
y_true = []
y_pred = []

# Melakukan evaluasi menggunakan K-Fold Cross Validation
for fold, (train_index, test_index) in enumerate(kf.split(X_pca, y), 1):
    X_train, X_test = X_pca.iloc[train_index], X_pca.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    best_svm_model.fit(X_train, y_train)
    y_pred_fold = best_svm_model.predict(X_test)

    y_true.extend(y_test)
    y_pred.extend(y_pred_fold)

# Menampilkan laporan klasifikasi
print("\nLaporan Klasifikasi SVM:")
print(classification_report(y_true, y_pred))

# Membuat confusion matrix
cm = confusion_matrix(y_true, y_pred)
print("\nConfusion Matrix:")
print(cm)
```

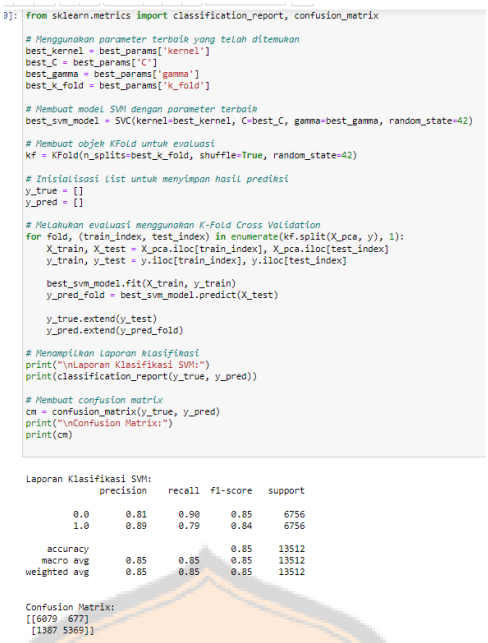
```
Laporan Klasifikasi SVM:
              precision    recall  f1-score   support

         0.0       0.81      0.91      0.85      6756
         1.0       0.90      0.78      0.83      6756

    accuracy                           0.84     13512
   macro avg       0.85      0.84      0.84     13512
weighted avg       0.85      0.84      0.84     13512


Confusion Matrix:
[[6142  614]
 [1482 5274]]
```

Gambar 4. 23 Confusion matrix dengan 3 fitur

```
]: from sklearn.metrics import classification_report, confusion_matrix

   # Menggunakan parameter terbaik yang telah ditemukan
   best_kernel = best_params['kernel']
   best_C = best_params['C']
   best_gamma = best_params['gamma']
   best_k_fold = best_params['k_fold']

   # Membuat model SVM dengan parameter terbaik
   best_svm_model = SVC(kernel=best_kernel, C=best_C, gamma=best_gamma, random_state=42)

   # Membuat objek KFold untuk evaluasi
   kf = KFold(n_splits=best_k_fold, shuffle=True, random_state=42)

   # Inisialisasi list untuk menyimpan hasil prediksi
   y_true = []
   y_pred = []

   # Melakukan evaluasi menggunakan K-Fold Cross Validation
   for fold, (train_index, test_index) in enumerate(kf.split(X_pca, y), 1):
       X_train, X_test = X_pca.iloc[train_index], X_pca.iloc[test_index]
       y_train, y_test = y.iloc[train_index], y.iloc[test_index]

       best_svm_model.fit(X_train, y_train)
       y_pred_fold = best_svm_model.predict(X_test)

       y_true.extend(y_test)
       y_pred.extend(y_pred_fold)

   # Menampilkan laporan klasifikasi
   print("\nLaporan Klasifikasi SVM:")
   print(classification_report(y_true, y_pred))

   # Membuat confusion matrix
   cm = confusion_matrix(y_true, y_pred)
   print("\nConfusion Matrix:")
   print(cm)
```

```
Laporan Klasifikasi SVM:
              precision    recall  f1-score   support

         0.0       0.81      0.90      0.85      6756
         1.0       0.89      0.79      0.84      6756

    accuracy                           0.85     13512
   macro avg       0.85      0.85      0.85     13512
weighted avg       0.85      0.85      0.85     13512


Confusion Matrix:
[[6079  677]
 [1387 5369]]
```

Gambar 4. 24 Confusion matrix dengan 5 fitur



```
3]: from sklearn.metrics import classification_report, confusion_matrix

    # Menggunakan parameter terbaik yang telah ditemukan
    best_kernel = best_params['kernel']
    best_C = best_params['C']
    best_gamma = best_params['gamma']
    best_k_fold = best_params['k_fold']

    # Membuat model SVM dengan parameter terbaik
    best_svm_model = SVC(kernel=best_kernel, C=best_C, gamma=best_gamma, random_state=42)

    # Membuat objek KFold untuk evaluasi
    kf = KFold(n_splits=best_k_fold, shuffle=True, random_state=42)

    # Inisialisasi list untuk menyimpan hasil prediksi
    y_true = []
    y_pred = []

    # Melakukan evaluasi menggunakan K-Fold Cross Validation
    for fold, (train_index, test_index) in enumerate(kf.split(X_pca, y), 1):
        X_train, X_test = X_pca.iloc[train_index], X_pca.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]

        best_svm_model.fit(X_train, y_train)
        y_pred_fold = best_svm_model.predict(X_test)

        y_true.extend(y_test)
        y_pred.extend(y_pred_fold)

    # Menampilkan Laporan Klasifikasi
    print("\nLaporan Klasifikasi SVM:")
    print(classification_report(y_true, y_pred))

    # Membuat confusion matrix
    cm = confusion_matrix(y_true, y_pred)
    print("\nConfusion Matrix:")
    print(cm)

Laporan Klasifikasi SVM:
              precision    recall  f1-score   support

         0.0       0.81      0.90      0.85      6756
         1.0       0.89      0.79      0.84      6756

    accuracy                           0.85     13512
   macro avg       0.85      0.85      0.85     13512
weighted avg       0.85      0.85      0.85     13512


Confusion Matrix:
[[6079  677]
 [1387 5369]]
```

Gambar 4. 25 Confusion matrix dengan 7 fitur

```
from sklearn.metrics import classification_report, confusion_matrix

# Menggunakan parameter terbaik yang telah ditemukan
best_kernel = best_params['kernel']
best_C = best_params['C']
best_gamma = best_params['gamma']
best_k_fold = best_params['k_fold']

# Membuat model SVM dengan parameter terbaik
best_svm_model = SVC(kernel=best_kernel, C=best_C, gamma=best_gamma, random_state=42)

# Membuat objek KFold untuk evaluasi
kf = KFold(n_splits=best_k_fold, shuffle=True, random_state=42)

# Inisialisasi list untuk menyimpan hasil prediksi
y_true = []
y_pred = []

# Melakukan evaluasi menggunakan K-Fold Cross Validation
for fold, (train_index, test_index) in enumerate(kf.split(X_pca, y), 1):
    X_train, X_test = X_pca.iloc[train_index], X_pca.iloc[test_index]
    y_train, y_test = y.iloc[train_index], y.iloc[test_index]

    best_svm_model.fit(X_train, y_train)
    y_pred_fold = best_svm_model.predict(X_test)

    y_true.extend(y_test)
    y_pred.extend(y_pred_fold)

# Menampilkan Laporan Klasifikasi
print("\nLaporan Klasifikasi SVM:")
print(classification_report(y_true, y_pred))

# Membuat confusion matrix
cm = confusion_matrix(y_true, y_pred)
print("\nConfusion Matrix:")
print(cm)

Laporan Klasifikasi SVM:
              precision    recall  f1-score   support

         0.0       0.81      0.90      0.85      6756
         1.0       0.89      0.79      0.84      6756

    accuracy                           0.85     13512
   macro avg       0.85      0.85      0.85     13512
weighted avg       0.85      0.85      0.85     13512


Confusion Matrix:
[[6079  677]
 [1387 5369]]
```

Gambar 4. 26 Confusion matrix dengan 9 fitur

4.7 Implementasi Pengujian

Pada tahap ini, dilakukan pengujian terhadap sistem klasifikasi untuk menemukan performa SVM terbaik dalam menangani kasus memprediksi employee turnover. Pengujian dilakukan dengan mengubah variasi feature selection, variasi fitur, variasi pada parameter, dan variasi pada nilai K-fold.

1. Pengujian pertama

Pada percobaan pertama dilakukan dengan menngunkan 3 fitur dan kombinasi Karnel,C ,Gamma, dengan menngunkan kombinasi K. hasil mendapatkan kernel linear memiliki akurasi sekitar 66.3% untuk semua kombinasi nilai C dan gamma. Hal ini menandakan bahwa model tersebut tidak terlalu sensitif terhadap perubahan parameter. Namun, pada model dengan kernel rbf, akurasinya meningkat hingga mencapai 75.5% saat menggunakan C=10, gamma=9, dan K-Fold=9. Di sisi lain, kernel sigmoid menunjukkan kinerja yang jauh lebih rendah dengan akurasi sekitar 51.2% hingga 54.0%.

Tabel 4. 1Kombinasi parameter dengan 3 fitur

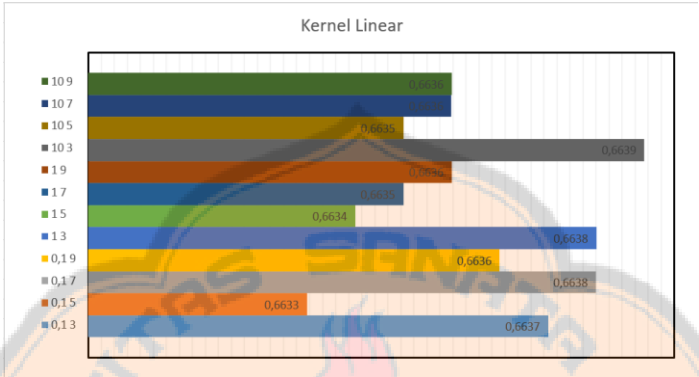
3 Fitur				
Karnel	C	Gamma	K	Akurasi
linear	0.1	0.1	3	0.6637063351095324
linear	0.1	0.1	5	0.6633361018666925
linear	0.1	0.1	7	0.6637799416662418
linear	0.1	0.1	9	0.663632096529029
linear	0.1	1.0	3	0.6637063351095324
linear	0.1	1.0	5	0.6633361018666925
linear	0.1	1.0	7	0.6637799416662418
linear	0.1	1.0	9	0.663632096529029
linear	0.1	10	3	0.6637063351095324

linear	0.1	10	5	0.6633361018666925
linear	0.1	10	7	0.6637799416662418
linear	0.1	10	9	0.663632096529029
linear	1.0	0.1	3	0.6637803433984607
linear	1.0	0.1	5	0.6634100937275879
linear	1.0	0.1	7	0.663483864686227
linear	1.0	0.1	9	0.6635581703729792
linear	1.0	1.0	3	0.6637803433984607
linear	1.0	1.0	5	0.6634100937275879
linear	1.0	1.0	7	0.663483864686227
linear	1.0	1.0	9	0.6635581703729792
linear	1.0	10	3	0.6637803433984607
linear	1.0	10	5	0.6634100937275879
linear	1.0	10	7	0.663483864686227
linear	1.0	10	9	0.6635581703729792
linear	10	0.1	3	0.663854351687389
linear	10	0.1	5	0.6634841129725915
linear	10	0.1	7	0.6635578839312307
linear	10	0.1	9	0.6635581703729792
linear	10	1.0	3	0.663854351687389
linear	10	1.0	5	0.6634841129725915
linear	10	1.0	7	0.6635578839312307
linear	10	1.0	9	0.6635581703729792
linear	10	10	3	0.663854351687389
linear	10	10	5	0.6634841129725915
linear	10	10	7	0.6635578839312307
linear	10	10	9	0.6635581703729792
rbf	0.1	0.1	3	0.7095914742451154
rbf	0.1	0.1	5	0.710553616304279
rbf	0.1	0.1	7	0.7104058945538165
rbf	0.1	0.1	9	0.7109247777705823

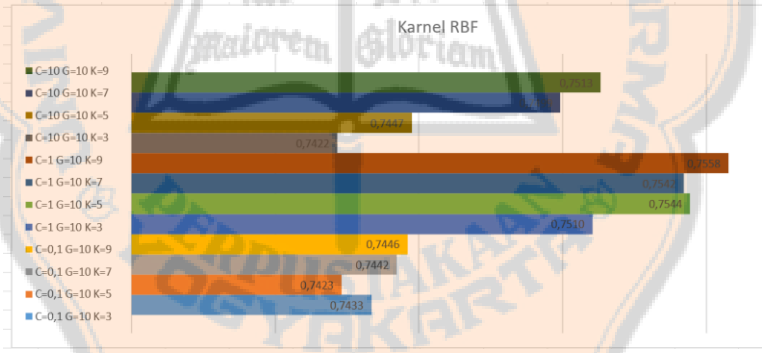
rbf	0.1	1.0	3	0.7369745411486086
rbf	0.1	1.0	5	0.7381590156837003
rbf	0.1	1.0	7	0.7376409135147796
rbf	0.1	1.0	9	0.7389002094475853
rbf	0.1	10	3	0.7433392539964476
rbf	0.1	10	5	0.7423029158872465
rbf	0.1	10	7	0.7442270547048603
rbf	0.1	10	9	0.7445975504420339
rbf	1.0	0.1	3	0.71751036116045
rbf	1.0	0.1	5	0.7178063795661974
rbf	1.0	0.1	7	0.7200270164494452
rbf	1.0	0.1	9	0.7196577238688534
rbf	1.0	1.0	3	0.7433392539964476
rbf	1.0	1.0	5	0.7434874154960645
rbf	1.0	1.0	7	0.7440059169397509
rbf	1.0	1.0	9	0.7437848555665262
rbf	1.0	10	3	0.7510361160449971
rbf	1.0	10	5	0.7544404016372411
rbf	1.0	10	7	0.7542184644859036
rbf	1.0	10	9	0.7557723760428393
rbf	10	0.1	3	0.7282415630550622
rbf	10	0.1	5	0.7274275669794753
rbf	10	0.1	7	0.7276500403828455
rbf	10	0.1	9	0.728317335220323
rbf	10	1.0	3	0.7440053285968028
rbf	10	1.0	5	0.7444497204493293
rbf	10	1.0	7	0.7454853435378439
rbf	10	1.0	9	0.7441542399262552
rbf	10	10	3	0.7421551213735939
rbf	10	10	5	0.7447457426611275
rbf	10	10	7	0.7499255782681643

rbf	10	10	9	0.7513324696786943
sigmoid	0.1	0.1	3	0.6195233866193014
sigmoid	0.1	0.1	5	0.6073118033996275
sigmoid	0.1	0.1	7	0.6015387263246704
sigmoid	0.1	0.1	9	0.5992446816784077
sigmoid	0.1	1.0	3	0.5270870337477799
sigmoid	0.1	1.0	5	0.5253849041816355
sigmoid	0.1	1.0	7	0.5247195912573727
sigmoid	0.1	1.0	9	0.5250154283887676
sigmoid	0.1	10	3	0.5243487270574304
sigmoid	0.1	10	5	0.5182074198337073
sigmoid	0.1	10	7	0.5216100163256325
sigmoid	0.1	10	9	0.5123594676094518
sigmoid	1.0	0.1	3	0.5461071640023683
sigmoid	1.0	0.1	5	0.5491421517282247
sigmoid	1.0	0.1	7	0.5447754717624822
sigmoid	1.0	0.1	9	0.5452936194137578
sigmoid	1.0	1.0	3	0.5246447602131439
sigmoid	1.0	1.0	5	0.5255326688305589
sigmoid	1.0	1.0	7	0.5233134172626986
sigmoid	1.0	1.0	9	0.5249415022327177
sigmoid	1.0	10	3	0.5247187685020722
sigmoid	1.0	10	5	0.5188737025751742
sigmoid	1.0	10	7	0.5193152280701216
sigmoid	1.0	10	9	0.5199805643207335
sigmoid	10	0.1	3	0.5401865008880994
sigmoid	10	0.1	5	0.540408428499956
sigmoid	10	0.1	7	0.5401127959763584
sigmoid	10	0.1	9	0.5391506022665365
sigmoid	10	1.0	3	0.5245707519242155
sigmoid	10	1.0	5	0.5240527768444361

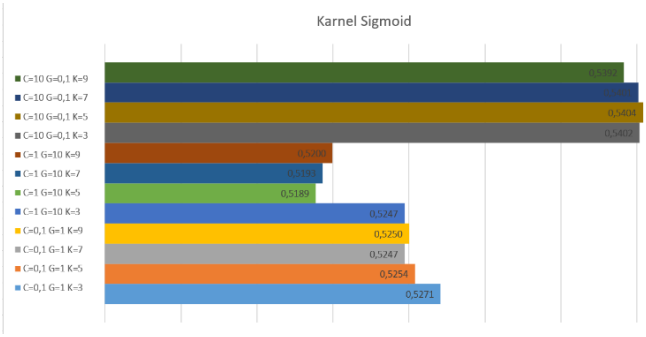
sigmoid	10	1.0	7	0.5239053412302528
sigmoid	10	1.0	9	0.5250894531130255
sigmoid	10	10	3	0.5231645944345767
sigmoid	10	10	5	0.5187996559460621
sigmoid	10	10	7	0.5188711509321787
sigmoid	10	10	9	0.520794244878322



Gambar 4. 27 Grafik karnel linear dengan 3 fitur



Gambar 4. 28 Grafik karnel RBF dengan 3 fitur



Gambar 4. 29 Grafik karnel Sigmoid dengan 3 fitur

2. Pengujian Kedua

Pada percobaan kedua dengan 5 fitur, pada percobaan ini menunjukan konsistensi kernel linear dalam akurasi sekitar 66.8% untuk semua kombinasi nilai C dan gamma, menunjukkan ketidak sensitifan terhadap perubahan parameter.sedangkan kernel sigmoid masih sama sepert percobaan sebelmnya dengan tetap menunjukkan kinerja rendah, dengan akurasi berkisar antara 48.6% hingga 50.2%, berbeda dengan karnel rbf yang menunjukan performa yang meningkat dari percobaan sebelumnya dengan nilai akurasi mencapai 84.0% saat C=10, gamma=10, dan K-Fold=9.

Tabel 4. 2 Kombinasi parameter dengan 5 fitur

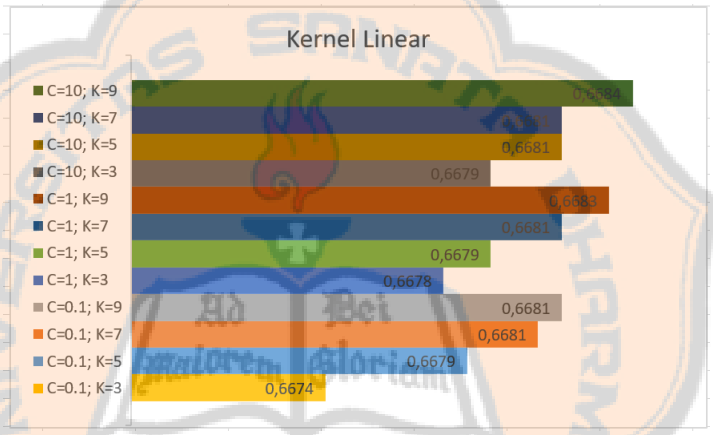
5 fitur				
Karnel	C	Gamma	K	Akurasi
linear	0.1	0.1	3	0.6674067495559504
linear	0.1	0.1	5	0.6678507555138586
linear	0.1	0.1	7	0.668072981212298
linear	0.1	0.1	9	0.6681473090041368
linear	0.1	1.0	3	0.6674067495559504

linear	0.1	1.0	5	0.6678507555138586
linear	0.1	1.0	7	0.668072981212298
linear	0.1	1.0	9	0.6681473090041368
linear	0.1	10	3	0.6674067495559504
linear	0.1	10	5	0.6678507555138586
linear	0.1	10	7	0.668072981212298
linear	0.1	10	9	0.6681473090041368
linear	1.0	0.1	3	0.6677767910005921
linear	1.0	0.1	5	0.6679247747588624
linear	1.0	0.1	7	0.66814707712146
linear	1.0	0.1	9	0.6682952598844446
linear	1.0	1.0	3	0.6677767910005921
linear	1.0	1.0	5	0.6679247747588624
linear	1.0	1.0	7	0.66814707712146
linear	1.0	1.0	9	0.6682952598844446
linear	1.0	10	3	0.6677767910005921
linear	1.0	10	5	0.6679247747588624
linear	1.0	10	7	0.66814707712146
linear	1.0	10	9	0.6682952598844446
linear	10	0.1	3	0.6679248075784487
linear	10	0.1	5	0.668146805109765
linear	10	0.1	7	0.6681470771214603
linear	10	0.1	9	0.6683692353245986
linear	10	1.0	3	0.6679248075784487
linear	10	1.0	5	0.668146805109765
linear	10	1.0	7	0.6681470771214603
linear	10	1.0	9	0.6683692353245986
linear	10	10	3	0.6679248075784487
linear	10	10	5	0.668146805109765
linear	10	10	7	0.6681470771214603
linear	10	10	9	0.6683692353245986

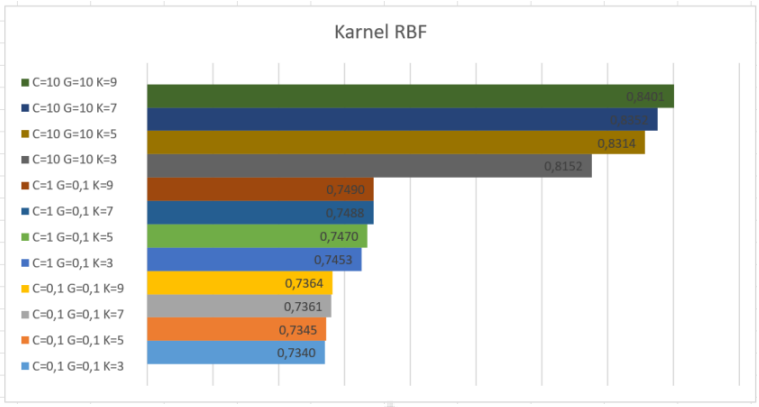
rbf	0.1	0.1	3	0.7340142095914742
rbf	0.1	0.1	5	0.7345317714533266
rbf	0.1	0.1	7	0.7360866626980186
rbf	0.1	0.1	9	0.736382629561256
rbf	0.1	1.0	3	0.7637655417406749
rbf	0.1	1.0	5	0.7674649955788357
rbf	0.1	1.0	7	0.7667989378947485
rbf	0.1	1.0	9	0.7686487353353918
rbf	0.1	10	3	0.6001332149200711
rbf	0.1	10	5	0.701601723884392
rbf	0.1	10	7	0.7226934725452233
rbf	0.1	10	9	0.7202510059132055
rbf	1.0	0.1	3	0.745263469508585
rbf	1.0	0.1	5	0.7469647317329512
rbf	1.0	0.1	7	0.748815634581822
rbf	1.0	0.1	9	0.7489633827978173
rbf	1.0	1.0	3	0.7874481941977501
rbf	1.0	1.0	5	0.789742351139302
rbf	1.0	1.0	7	0.7884095675336489
rbf	1.0	1.0	9	0.790112356914693
rbf	1.0	10	3	0.8168294849023091
rbf	1.0	10	5	0.8299294338910655
rbf	1.0	10	7	0.8314083090914874
rbf	1.0	10	9	0.8363673613458267
rbf	10	0.1	3	0.7571047957371225
rbf	10	0.1	5	0.757030007232143
rbf	10	0.1	7	0.7589542778792087
rbf	10	0.1	9	0.7595461584568715
rbf	10	1.0	3	0.7967732386027236
rbf	10	1.0	5	0.8054326511130407
rbf	10	1.0	7	0.806468653367224

rbf	10	1.0	9	0.8045446646556782
rbf	10	10	3	0.8152013025458852
rbf	10	10	5	0.8314095723341639
rbf	10	10	7	0.8351831755904386
rbf	10	10	9	0.8401411930439627
sigmoid	0.1	0.1	3	0.5787448194197751
sigmoid	0.1	0.1	5	0.5664601357211181
sigmoid	0.1	0.1	7	0.5654969888235156
sigmoid	0.1	0.1	9	0.5615769888181268
sigmoid	0.1	1.0	3	0.4906009473060983
sigmoid	0.1	1.0	5	0.4921564245993637
sigmoid	0.1	1.0	7	0.4919348155326185
sigmoid	0.1	1.0	9	0.4917876911767358
sigmoid	0.1	10	3	0.4877886323268206
sigmoid	0.1	10	5	0.48653175611822597
sigmoid	0.1	10	7	0.4868277559519178
sigmoid	0.1	10	9	0.48697721763436497
sigmoid	1.0	0.1	3	0.503922439313203
sigmoid	1.0	0.1	5	0.5024429910785313
sigmoid	1.0	0.1	7	0.5005187096962145
sigmoid	1.0	0.1	9	0.5017778254847904
sigmoid	1.0	1.0	3	0.4906009473060983
sigmoid	1.0	1.0	5	0.4918602654670236
sigmoid	1.0	1.0	7	0.49186079628761487
sigmoid	1.0	1.0	9	0.4913434935470844
sigmoid	1.0	10	3	0.4888247483718176
sigmoid	1.0	10	5	0.486235733906428
sigmoid	1.0	10	7	0.48645785138729547
sigmoid	1.0	10	9	0.4860151919236364
sigmoid	10	0.1	3	0.49962995855535813
sigmoid	10	0.1	5	0.49652235515381243

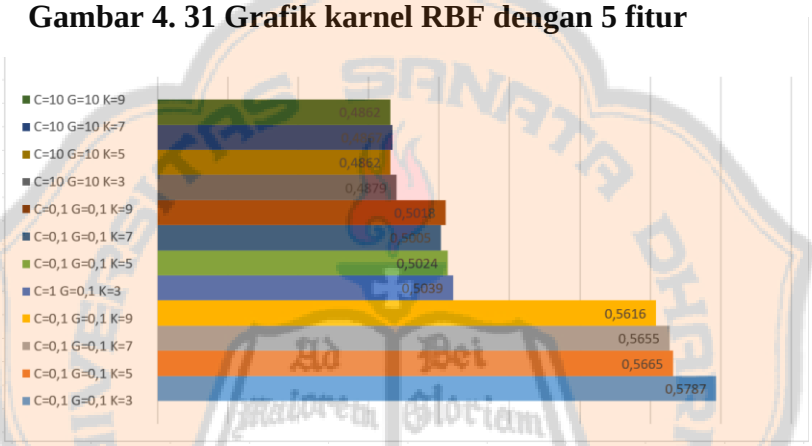
sigmoid	10	0.1	7	0.49807645793188465
sigmoid	10	0.1	9	0.49948315760100354
sigmoid	10	1.0	3	0.4903789224393132
sigmoid	10	1.0	5	0.4914164238380854
sigmoid	10	1.0	7	0.49178685370676956
sigmoid	10	1.0	9	0.49134359211529244
sigmoid	10	10	3	0.48786264061574897
sigmoid	10	10	5	0.4861616598932074
sigmoid	10	10	7	0.4866798324581481
sigmoid	10	10	9	0.4861632413721522



Gambar 4. 30 Grafik karnel Linear dengan 5 fitur



Gambar 4. 31 Grafik karnel RBF dengan 5 fitur



Gambar 4. 32 Grafik karnel Sigmoid dengan 5 fitur

3. Pengujian ketiga

Pada percobaan ketiga dengan menggunakan 7 fitur, hasilnya menunjukkan pola yang mirip dengan percobaan sebelumnya.Kernel linear tetap menunjukkan konsistensi dalam akurasi sekitar 66.8% untuk semua kombinasi nilai C dan gamm.Namun, kernel rbf menunjukkan peningkatan yang signifikan dalam akurasi, mencapai 84.7% saat menggunakan C=10, gamma=10, dan K-Fold=9. Di sisi lain,kernel sigmoid tetap menunjukkan kinerja yang rendah, dengan akurasi berkisar antara 47.7% hingga 49.4%.

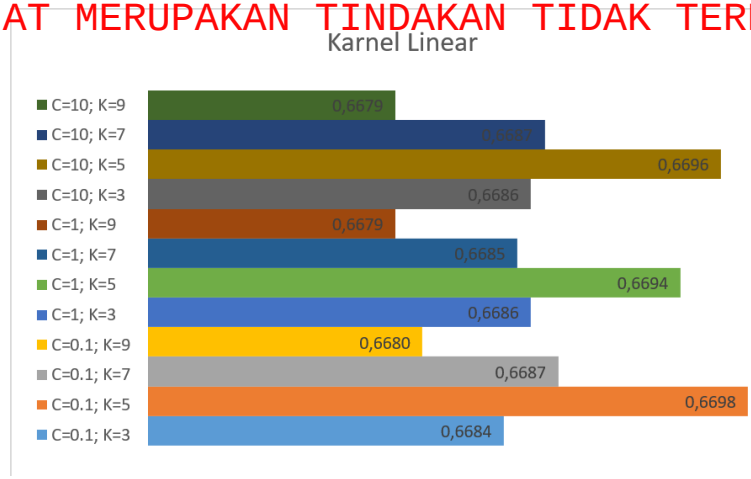
Tabel 4. 3 Kombinasi parameter dengan 7 fitur

Fitur 7				
Karnel	C	Gamma	K	Akurasi
linear	0.1	0.1	3	0.6684428656009475
linear	0.1	0.1	5	0.6697745986653534
linear	0.1	0.1	7	0.6687388094286183
linear	0.1	0.1	9	0.667999111703309
linear	0.1	1.0	3	0.6684428656009475
linear	0.1	1.0	5	0.6697745986653534
linear	0.1	1.0	7	0.6687388094286183
linear	0.1	1.0	9	0.667999111703309
linear	0.1	10	3	0.6684428656009475
linear	0.1	10	5	0.6697745986653534
linear	0.1	10	7	0.6687388094286183
linear	0.1	10	9	0.667999111703309
linear	1.0	0.1	3	0.6685908821788041
linear	1.0	0.1	5	0.6694046119767685
linear	1.0	0.1	7	0.6685167900256863
linear	1.0	0.1	9	0.6678509636865849
linear	1.0	1.0	3	0.6685908821788041
linear	1.0	1.0	5	0.6694046119767685
linear	1.0	1.0	7	0.6685167900256863
linear	1.0	1.0	9	0.6678509636865849
linear	1.0	10	3	0.6685908821788041
linear	1.0	10	5	0.6694046119767685
linear	1.0	10	7	0.6685167900256863
linear	1.0	10	9	0.6678509636865849
linear	10	0.1	3	0.6685908821788041

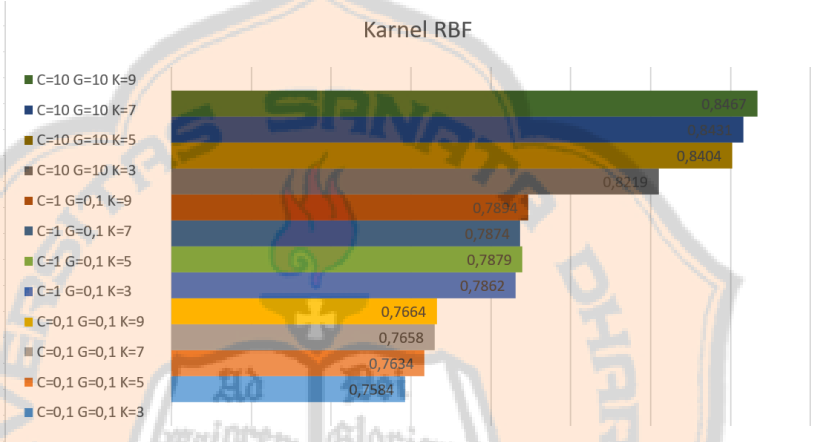
linear	10	0.1	5	0.6696265875594544
linear	10	0.1	7	0.6686647901836145
linear	10	0.1	9	0.6678510622547931
linear	10	1.0	3	0.6685908821788041
linear	10	1.0	5	0.6696265875594544
linear	10	1.0	7	0.6686647901836145
linear	10	1.0	9	0.6678510622547931
linear	10	10	3	0.6685908821788041
linear	10	10	5	0.6696265875594544
linear	10	10	7	0.6686647901836145
linear	10	10	9	0.6678510622547931
rbf	0.1	0.1	3	0.758436944937833
rbf	0.1	0.1	5	0.7633954706137025
rbf	0.1	0.1	7	0.7658379910003944
rbf	0.1	0.1	9	0.7664305563810644
rbf	0.1	1.0	3	0.7941829484902309
rbf	0.1	1.0	5	0.7999543780754065
rbf	0.1	1.0	7	0.8015084056499951
rbf	0.1	1.0	9	0.8015084188585033
rbf	0.1	10	3	0.5230905861456483
rbf	0.1	10	5	0.5887354100893462
rbf	0.1	10	7	0.7183340418379311
rbf	0.1	10	9	0.6699305064562668
rbf	1.0	0.1	3	0.786190053285968
rbf	1.0	0.1	5	0.7878918700142096
rbf	1.0	0.1	7	0.7873733747677555
rbf	1.0	0.1	9	0.789372405376738
rbf	1.0	1.0	3	0.8251924215512138
rbf	1.0	1.0	5	0.8319269129100462
rbf	1.0	1.0	7	0.8320744439644416
rbf	1.0	1.0	9	0.8316307646754015

rbf	1.0	10	3	0.8190497335701599
rbf	1.0	10	5	0.8376997294176249
rbf	1.0	10	7	0.8415472973775875
rbf	1.0	10	9	0.844878085419012
rbf	10	0.1	3	0.7975873297809355
rbf	10	0.1	5	0.7982529212682239
rbf	10	0.1	7	0.8006210946798523
rbf	10	0.1	9	0.7991410076371633
rbf	10	1.0	3	0.8311130846654825
rbf	10	1.0	5	0.843916195865383
rbf	10	1.0	7	0.8456179725319987
rbf	10	1.0	9	0.8428069209470157
rbf	10	10	3	0.8219360568383659
rbf	10	10	5	0.8404378116482686
rbf	10	10	7	0.8431017015226653
rbf	10	10	9	0.8466542845283693
sigmoid	0.1	0.1	3	0.572380106571936
sigmoid	0.1	0.1	5	0.5647573918608406
sigmoid	0.1	0.1	7	0.5572837658661268
sigmoid	0.1	0.1	9	0.5578752104800873
sigmoid	0.1	1.0	3	0.4786856127886323
sigmoid	0.1	1.0	5	0.479426894425773
sigmoid	0.1	1.0	7	0.4814995202740283
sigmoid	0.1	1.0	9	0.47905785352547425
sigmoid	0.1	10	3	0.48016577856719955
sigmoid	0.1	10	5	0.48431164429795776
sigmoid	0.1	10	7	0.47794621319305836
sigmoid	0.1	10	9	0.4799451645344885
sigmoid	1.0	0.1	3	0.4923031379514506
sigmoid	1.0	0.1	5	0.4897885755142804
sigmoid	1.0	0.1	7	0.48882604557454234

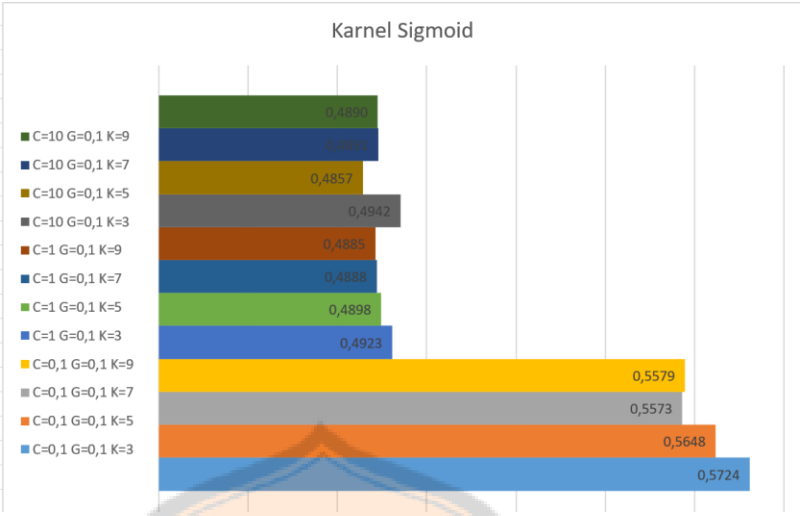
sigmoid	1.0	0.1	9	0.4885308004458044
sigmoid	1.0	1.0	3	0.48075784487862644
sigmoid	1.0	1.0	5	0.478612929187708
sigmoid	1.0	1.0	7	0.4814994436098699
sigmoid	1.0	1.0	9	0.4820170189839412
sigmoid	1.0	10	3	0.4799437537004145
sigmoid	1.0	10	5	0.4834235776625637
sigmoid	1.0	10	7	0.47728038497673814
sigmoid	1.0	10	9	0.479426794328267
sigmoid	10	0.1	3	0.4941533451746596
sigmoid	10	0.1	5	0.4857178456483776
sigmoid	10	0.1	7	0.48912166256960626
sigmoid	10	0.1	9	0.4890473471401765
sigmoid	10	1.0	3	0.4809058614564831
sigmoid	10	1.0	5	0.4787609129094985
sigmoid	10	1.0	7	0.48112950071316835
sigmoid	10	1.0	9	0.48275721694241613
sigmoid	10	10	3	0.48031379514505623
sigmoid	10	10	5	0.4834235776625637
sigmoid	10	10	7	0.477280346644659
sigmoid	10	10	9	0.47964896706924876



Gambar 4. 33 Grafik karnel Linear dengan 7 fitur



Gambar 4. 34 Grafik karnel RBF dengan 7 fitur



Gambar 4. 35 Grafik karnel Sigmoid dengan 7 fitur

4. Pengujian Keempat

Pada percobaan keempat dengan menggunakan 9 fitur, pola yang mirip dengan percobaan sebelumnya masih dapat diamati. Kernel linear menunjukkan konsistensi dalam akurasi sekitar 66.8% untuk semua kombinasi nilai C dan gamma. Kernel rbf,menunjukkan peningkatan dalam akurasi, mencapai 84.7% saat menggunakan C=10, gamma=10, dan K-Fold=9. Hal ini menandakan bahwa dengan penambahan fitur, kernel rbf masih mampu menangkap pola yang lebih kompleks dalam data, dan meningkatkan kinerjanya secara signifikan. Namun, kernel sigmoid tetap menunjukkan kinerja yang rendah, dengan akurasi berkisar antara 49.0% hingga 50.5%, menegaskan ketidakcocokannya untuk menangani pola data yang kompleks, terutama setelah penambahan fitur.

Tabel 4. 4 Kombinasi parameter dengan 9 fitur

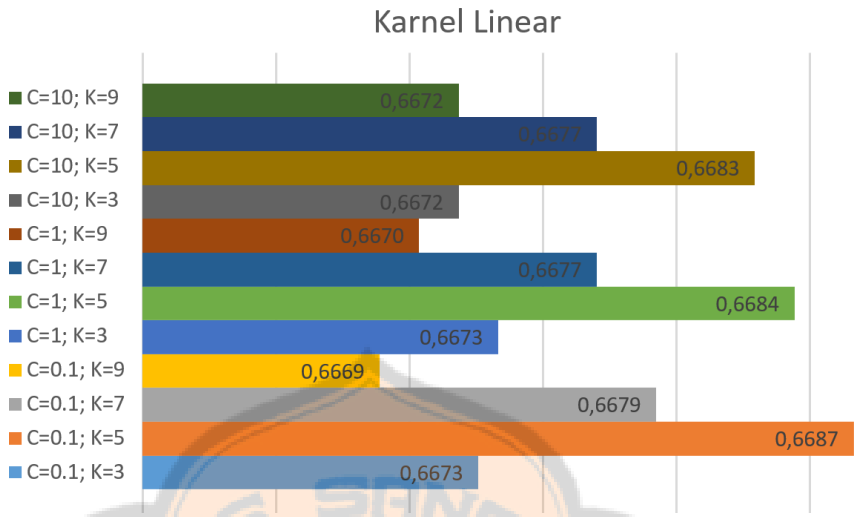
Fitur 9				
Karnel	C	Gamma	K	Akurasi
linear	0.1	0.1	3	0.6672587329780937
linear	0.1	0.1	5	0.6686645564472734
linear	0.1	0.1	7	0.6679246360656567
linear	0.1	0.1	9	0.6668888394076484
linear	0.1	1.0	3	0.6672587329780937
linear	0.1	1.0	5	0.6686645564472734
linear	0.1	1.0	7	0.6679246360656567
linear	0.1	1.0	9	0.6668888394076484
linear	0.1	10	3	0.6672587329780937
linear	0.1	10	5	0.6686645564472734
linear	0.1	10	7	0.6679246360656567
linear	0.1	10	9	0.6668888394076484
linear	1.0	0.1	3	0.6673327412670219
linear	1.0	0.1	5	0.6684425534804792

linear	1.0	0.1	7	0.6677026549948041
linear	1.0	0.1	9	0.6670368395720603
linear	1.0	1.0	3	0.6673327412670219
linear	1.0	1.0	5	0.6684425534804792
linear	1.0	1.0	7	0.6677026549948041
linear	1.0	1.0	9	0.6670368395720603
linear	1.0	10	3	0.6673327412670219
linear	1.0	10	5	0.6684425534804792
linear	1.0	10	7	0.6677026549948041
linear	1.0	10	9	0.6670368395720603
linear	10	0.1	3	0.6671847246891652
linear	10	0.1	5	0.6682945423745801
linear	10	0.1	7	0.6677026933268834
linear	10	0.1	9	0.667184839736472
linear	10	1.0	3	0.6671847246891652
linear	10	1.0	5	0.6682945423745801
linear	10	1.0	7	0.6677026933268834
linear	10	1.0	9	0.667184839736472
linear	10	10	3	0.6671847246891652
linear	10	10	5	0.6682945423745801
linear	10	10	7	0.6677026933268834
linear	10	10	9	0.667184839736472
rbf	0.1	0.1	3	0.753922439313203
rbf	0.1	0.1	5	0.7588069346420746
rbf	0.1	0.1	7	0.7608794298946518
rbf	0.1	0.1	9	0.7608794413232821
rbf	0.1	1.0	3	0.7645796329188869
rbf	0.1	1.0	5	0.7756056064032808
rbf	0.1	1.0	7	0.778269774273885
rbf	0.1	1.0	9	0.7780481503725039
rbf	0.1	10	3	0.5142835997631735

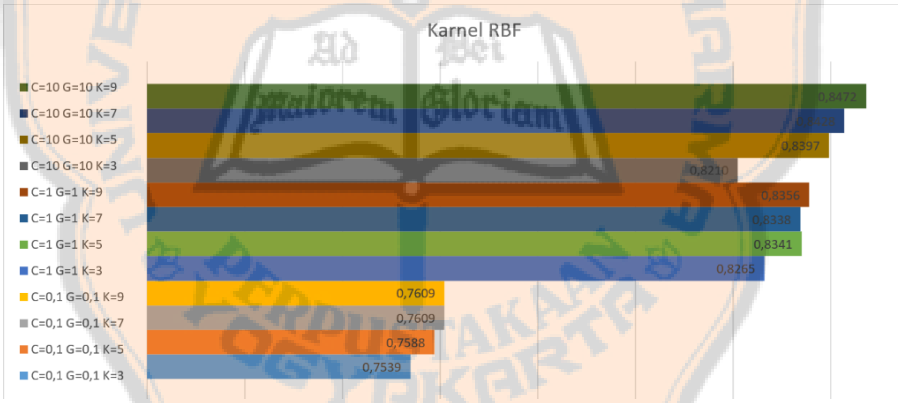
rbf	0.1	10	5	0.5647565429534802
rbf	0.1	10	7	0.6839152079074481
rbf	0.1	10	9	0.6326283045114965
rbf	1.0	0.1	3	0.7864860864416815
rbf	1.0	0.1	5	0.7888537915899569
rbf	1.0	0.1	7	0.7872995855152273
rbf	1.0	0.1	9	0.7892238138030778
rbf	1.0	1.0	3	0.826450562462996
rbf	1.0	1.0	5	0.8340735805515871
rbf	1.0	1.0	7	0.8337770782599229
rbf	1.0	1.0	9	0.8356275576601839
rbf	1.0	10	3	0.8177175843694494
rbf	1.0	10	5	0.834887052875701
rbf	1.0	10	7	0.8396230653320459
rbf	1.0	10	9	0.8448775925779717
rbf	10	0.1	3	0.8018798105387802
rbf	10	0.1	5	0.8032850250277057
rbf	10	0.1	7	0.804764907441445
rbf	10	0.1	9	0.8049131619015344
rbf	10	1.0	3	0.8322232089994079
rbf	10	1.0	5	0.844730763553833
rbf	10	1.0	7	0.8438423156255738
rbf	10	1.0	9	0.8440647498501517
rbf	10	10	3	0.8209739490822971
rbf	10	10	5	0.8396975918141232
rbf	10	10	7	0.8428055478784919
rbf	10	10	9	0.8472458909131841
sigmoid	0.1	0.1	3	0.5800029603315572
sigmoid	0.1	0.1	5	0.5646099284371096
sigmoid	0.1	0.1	7	0.5630563086744346
sigmoid	0.1	0.1	9	0.5580244427470998

sigmoid	0.1	1.0	3	0.49896388395500296
sigmoid	0.1	1.0	5	0.49778046324600816
sigmoid	0.1	1.0	7	0.4977052500765683
sigmoid	0.1	1.0	9	0.49445041274944285
sigmoid	0.1	10	3	0.5019982238010657
sigmoid	0.1	10	5	0.4943029553203626
sigmoid	0.1	10	7	0.5045917997716175
sigmoid	0.1	10	9	0.4953388572928498
sigmoid	1.0	0.1	3	0.4965216104203671
sigmoid	1.0	0.1	5	0.49237703097662955
sigmoid	1.0	0.1	7	0.4879373163174678
sigmoid	1.0	0.1	9	0.48919574157742063
sigmoid	1.0	1.0	3	0.4984458259325044
sigmoid	1.0	1.0	5	0.49378331447937474
sigmoid	1.0	1.0	7	0.5034050002664079
sigmoid	1.0	1.0	9	0.4962998973214976
sigmoid	1.0	10	3	0.5013321492007105
sigmoid	1.0	10	5	0.4960787873659582
sigmoid	1.0	10	7	0.503925434906188
sigmoid	1.0	10	9	0.49726394368049154
sigmoid	10	0.1	3	0.4937092954410894
sigmoid	10	0.1	5	0.49430243502230303
sigmoid	10	0.1	7	0.49037975974219383
sigmoid	10	0.1	9	0.48830818414788624
sigmoid	10	1.0	3	0.4993339253996448
sigmoid	10	1.0	5	0.4970401064913207
sigmoid	10	1.0	7	0.5014057906738818
sigmoid	10	1.0	9	0.49852073761744287
sigmoid	10	10	3	0.5014061574896389
sigmoid	10	10	5	0.4963008724850777
sigmoid	10	10	7	0.5041475309732782

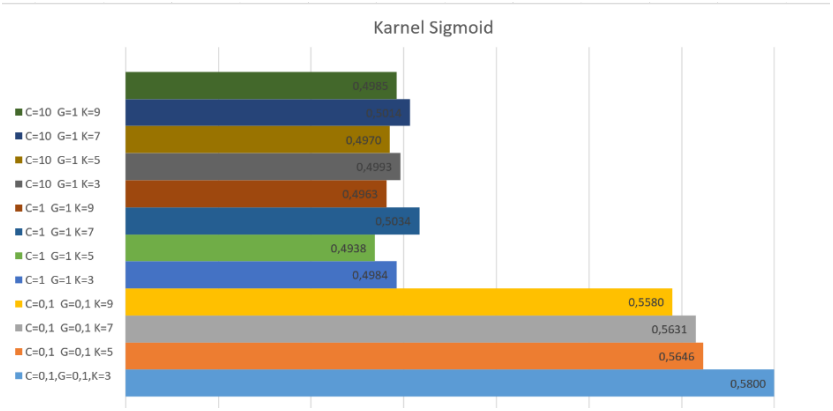
sigmoid	10	10	9	0.4974861657055773



Gambar 4. 36 Grafik karnel Linear dengan 9 fitur



Gambar 4. 37 Grafik karnel RBF dengan 9 fitur



Gambar 4. 38 Grafik karnel Sigmoid dengan 9 fitur

4.8 Analisis Hasil

Berdasarkan nilai akurasi yang diperoleh dari setiap pengujian, diperoleh Beberapa variasi terbaik

Tabel 4. 5 analisa hasil dari kombinasi terbaik

Fitur	kombinasi				
	Karnel	C	Gamma	K	Akurasi
3 Fitur	rbf	1.0	10	9	0.7557
5 Fitur	rbf	10	10	9	0.8401
7 Fitur	rbf	10	10	9	0.8466
9 Fitur	rbf	10	10	9	0.8472

Dari hasil berbagai percobaan yang dilakukan, diperoleh akurasi terbaik yaitu 84.72% dengan menggunakan kombinasi kernel "rbf", nilai C sebesar 10, dan gamma sebesar 10 dengan 9 fitur. Pada Tabel 4.5 menunjukkan bahwa menambah jumlah fitur dari 3 menjadi 9 memberikan peningkatan akurasi yang konsisten: dengan 3 fitur, akurasi yang diperoleh adalah 75.57%, dengan 5 fitur meningkat signifikan menjadi

84.01%, dengan 7 fitur sedikit meningkat lagi menjadi 84.66%, dan pada penggunaan 9 fitur, akurasi tertinggi tercapai yaitu 84.72%. Hal ini menunjukkan bahwa fitur tambahan memberikan informasi penting yang membantu model dalam membuat prediksi lebih akurat.

Penggunaan kernel "rbf" dengan nilai C sebesar 10 dan gamma sebesar 10 memberikan hasil terbaik pada setiap jumlah fitur yang diuji. Kombinasi parameter ini menghasilkan akurasi yang tinggi karena kernel "rbf" mampu menangani data yang tidak linier dengan baik, sementara nilai C dan gamma yang dipilih mengatur keseimbangan antara margin dan keakuratan model secara optimal.

Metode validasi yang digunakan dalam percobaan ini adalah k-fold cross validation dengan nilai k yang bervariasi, yaitu 3, 5, 7, dan 9. Hasil penelitian menunjukkan bahwa penggunaan k-fold dengan k=9 terbukti lebih baik dibandingkan dengan nilai k lainnya. Dengan k=9, model diuji secara menyeluruh dengan pembagian data yang bervariasi, sehingga hasil akurasinya lebih dapat diandalkan.

Secara keseluruhan, hasil percobaan menunjukkan bahwa untuk memprediksi employee turnover dengan model SVM, penggunaan kernel "rbf" dengan nilai C sebesar 10 dan gamma sebesar 10 serta 9 fitur memberikan akurasi terbaik. Pemilihan parameter ini sangat mempengaruhi kinerja model, dan balancing data serta jumlah fitur yang digunakan juga memainkan peran penting dalam hasil akhir yang diperoleh. Penggunaan k-fold cross validation dengan k=9 memastikan bahwa model diuji secara optimal, memberikan hasil yang lebih baik dibandingkan dengan nilai k lainnya.

BAB V
PENUTUP

Bab terakhir ini berisi kesimpulan dari seluruh penelitian dan saran diberikan sebagai panduan untuk pengembangan lebih lanjut.

5.1 Kesimpulan

Berdasarkan hasil dan analisis yang didapatkan dari penerapan Algoritma SVM dalam memprediksi *employee turnover* dengan menggunakan total data 9540, dapat disimpulkan beberapa hal, antara lain:

1. **Kemampuan Algoritma SVM dalam Prediksi Employee Turnover:** Algoritma Support Vector Machine (SVM) menunjukkan kemampuan yang baik dalam memprediksi employee turnover. Penggunaan kombinasi parameter kernel 'rbf' dengan nilai C=10 dan gamma=10 menghasilkan akurasi yang tinggi, yakni 84.72%. Akurasi ini merupakan yang terbaik dibandingkan dengan berbagai kombinasi parameter lainnya.
2. **Kombinasi Parameter Terbaik:** Dari hasil penelitian ini, kombinasi parameter terbaik pada algoritma SVM untuk memprediksi employee turnover ditemukan dengan kernel 'rbf', nilai C=10, dan nilai gamma=10. Kombinasi ini memberikan akurasi tertinggi sebesar 84.72%.
3. **Stabilitas Model dalam K-Fold Cross-Validation:** Hasil k-fold cross-validation menunjukkan bahwa model SVM tetap stabil dalam kinerjanya ketika diuji pada dataset yang dibagi menjadi bagian-bagian yang berbeda. Penggunaan k-fold dengan nilai k yang bervariasi, yaitu 3, 5, 7, dan 9, menunjukkan bahwa nilai k=9 memberikan hasil akurasi yang paling baik dengan nilai akurasi tertinggi 84.72%.

4. **Fungsi Penelitian untuk Perusahaan:** Penelitian ini memberikan dasar yang kuat bagi perusahaan untuk mengidentifikasi faktor-faktor yang mempengaruhi turnover dan merancang strategi retensi yang efektif. Dengan model prediktif yang akurat, perusahaan dapat mengambil langkah-langkah proaktif untuk mempertahankan karyawan yang berisiko tinggi meninggalkan perusahaan, sehingga mengurangi tingkat turnover dan meningkatkan stabilitas serta produktivitas tenaga kerja.

5.1 Saran

Berikut saran untuk mengembangkan penelitian ini :

1. Menambahkan penerapan algoritma klasifikasi selain SVM: Menguji algoritma klasifikasi lain seperti *Random Forest*, *Neural Networks*, atau *Gradient Boosting* untuk membandingkan kinerja dan akurasi prediksi.
2. Menambahkan parameter yang lain: Eksplorasi lebih lanjut pada parameter-parameter lain dalam SVM serta dalam algoritma klasifikasi lainnya untuk menemukan konfigurasi yang lebih optimal.
3. Menggunakan rentang data yang lebih banyak: Memperluas dataset dengan data karyawan dari periode waktu yang lebih panjang atau dari berbagai perusahaan dan industri untuk meningkatkan generalisasi model.
4. Menambahkan *Graphic User Interface* (GUI): Mengembangkan GUI untuk mempermudah proses input parameter yang digunakan saat percobaan, sehingga proses eksperimentasi dan evaluasi model menjadi lebih *user-friendly*.

DAFTAR PUSTAKA

[1] Price, J. (1977). *The study of turnover*. Iowa: Iowa State University Press.

[2] Neurocomputing. (2020, 30 September). A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408, hlm. 189-215 https://www.sciencedirect.com/science/article/abs/pii/S0925231220307153

[3] Employee turnover rates have increased by 9% since 2019. (7 Februari 2023). Diakses pada 5 Desember 2023, Diakses pada desember 2023 https://hrreview-co- uk.translate.goog/hr-news/recruitment/employee-turnover-rates-have-increased-by-9- since-2019/150788?_x_tr_sl=en&_x_tr_tl=id&_x_tr_hl=id&_x_tr_pto=tc.

[4] Sim, K. S., & Read, A. (2020). Support vector machine (SVM) in engineering applications. Dalam *Advances in Engineering Research and Application* (hlm. 307-315). Springer. https://link.springer.com/chapter/10.1007/978-3-030-36365-9_25

[5] Kesuma, R., Dinata, D., & Hasdyna, N. (2020). *Machine Learning* (Edisi pertama). ISBN 978-602-464-096-5.. https://repository.unimal.ac.id/6707/1/Machine%20Learning.pdf.

[6] DataCamp. (September 2022). Classification in machine learning. Diakses pada desember 2023 https://www-datacamp-com.translate.goog/blog/classification-machine-learning?_x_tr_sl=en&_x_tr_tl=id&_x_tr_hl=id&_x_tr_pto=tc&_x_tr_hist=true.

[7] Universitas Pembangunan Jaya. (2011, 25 Februari). Handout TIF311 DM-4. Diakses dari https://ocw.upj.ac.id/files/Handout-TIF311-DM-4.PDF.

[8] "What Is Feature Extraction?" . Diakses pada 6 Januari 2024, dari https://www.mathworks.com/discovery/feature-extraction.html.

[9] International Journal of Emerging Technology and Advanced Engineering. (2012, Agustus). SUPPORT VECTOR MACHINE-A Survey. Volume 2, Issue 8. Diakses dari www.ijetae.com (ISSN 2250-2459).

[10] Chen, X.-w., & Jeong, J. C. (January 2008). Enhanced Recursive Feature Elimination. Electrical Engineering and Computer Science Department, The University of Kansas. DOI: 10.1109/ICMLA.2007.35. Diakses dari IEEE Xplor.

[11] BambooHR. (n.d.). Employee turnover. Diakses pada 6 Januari 2024, dari https://www.bamboohr.com/resources/hr-glossary/employee-turnover

[12] Pisner, D., & Schnyer, D. M. (2020, Januari). Support vector machine. Dalam D. Pisner (Ed.), *Machine Learning* (hlm. 101-121). University of Texas at Austin. DOI: 10.1016/B978-0-12-815739-8.00006- https://www.researchgate.net/publication/338315621_Support_vector_machine.

[13] Pooja, S., & Kumar, D. (2020). Support vector machine. https://www.researchgate.net/publication/338315621_Support_vector_machine.