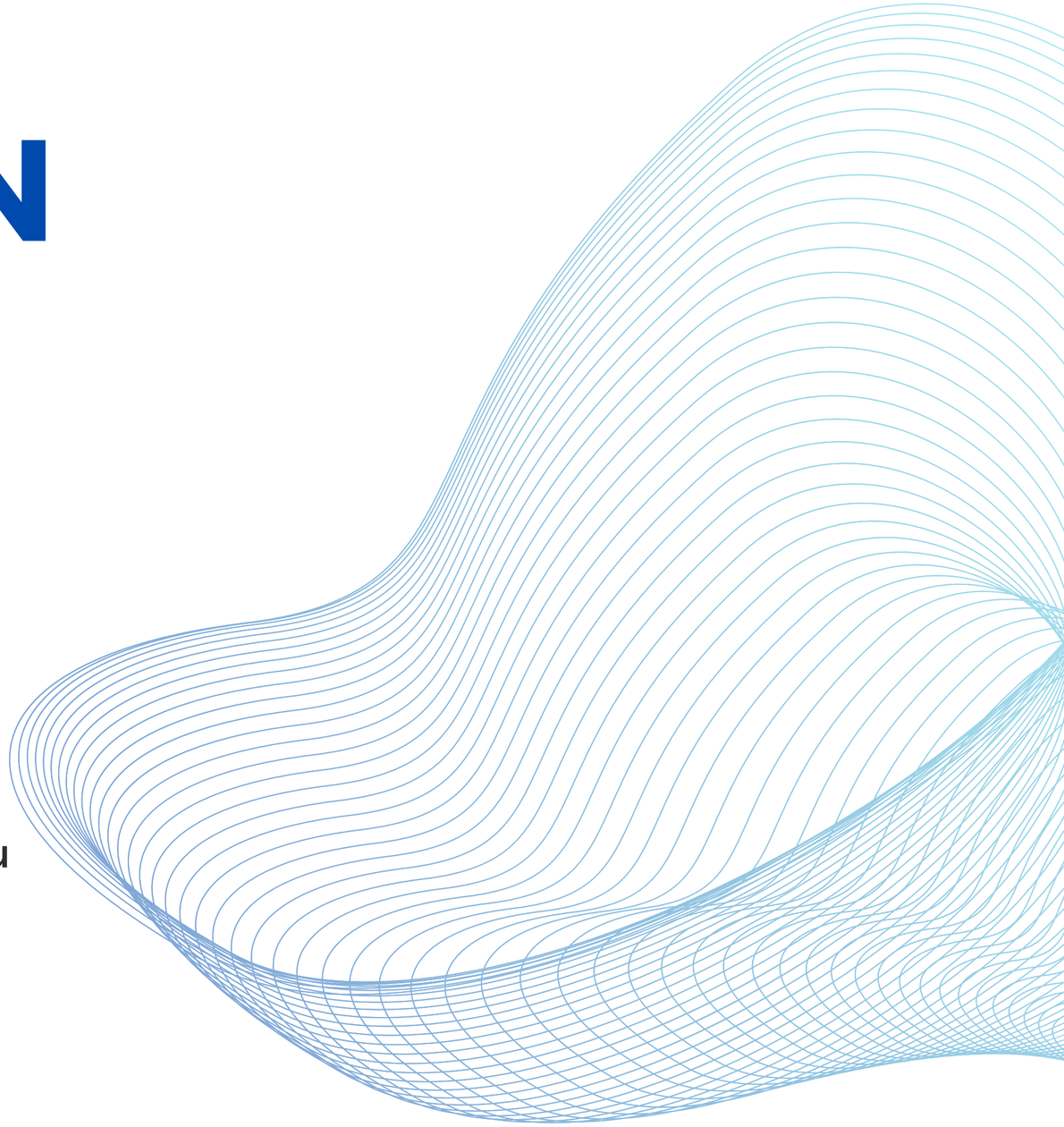
PENERAPAN SUPPORT VECTOR MACHINE (SVM) UNTUK KLASIFIKASI KANKER

DATA SCIENCE PROJECT FROM DIBIMBING.ID

Arjuna Muhammad Malik

PENDAHULUAN

Machine learning digunakan secara luas dalam analisis data medis untuk membantu dalam diagnosis dan prediksi penyakit. Salah satu algoritma yang efektif dalam klasifikasi adalah Support Vector Machine (SVM), yang mampu memisahkan data dengan optimal menggunakan hyperplane. Dalam studi ini, digunakan dataset Breast Cancer Wisconsin yang tersedia di scikit-learn, yang berisi informasi mengenai karakteristik tumor dan klasifikasinya sebagai malignant (ganas) atau benign (jinak). Dengan menerapkan model SVM, diharapkan dapat diperoleh hasil klasifikasi yang akurat untuk membantu dalam deteksi dini kanker payudara.



TAHAPAN

Dalam membangun model klasifikasi menggunakan Support Vector Machine (SVM) pada dataset Breast Cancer Wisconsin, terdapat beberapa tahapan utama yang harus dilakukan

01 Import Library

02 Load Dataset

03 Data Preprocessing

04 Split Data

05 Train the Model

06 Model Evaluation

07 Visualization

IMPORT LIBRARY

Pada tahap awal, berbagai pustaka seperti numpy, pandas, matplotlib, seaborn dan scikit-learn diimpor untuk mendukung pemrosesan data dan pembuatan model.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

LOAD DATASET

Dataset Breast Cancer Wisconsin diambil dari pustaka scikit-learn, kemudian dikonversi menjadi DataFrame pandas agar lebih mudah dianalisis.

```
# Memuat dataset Iris dari scikit-learn
cancer = datasets.load_breast_cancer()

X = cancer.data     # inputan untuk machine learning
y = cancer.target   # output yang diinginkan dari machine learning

# Mengonversi data fitur dan target menjadi DataFrame
df_X = pd.DataFrame(X, columns=cancer.feature_names)
df_y = pd.Series(y, name='target')
```

DATA PREPROCESSING

Melakukan eksplorasi data menggunakan df.describe() dan df.info() untuk melihat distribusi data serta pengecekan missing values. Jika diperlukan, data dapat dibersihkan atau diubah sesuai kebutuhan.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 31 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   mean radius             569 non-null    float64
 1   mean texture            569 non-null    float64
 2   mean perimeter          569 non-null    float64
 3   mean area               569 non-null    float64
 4   mean smoothness         569 non-null    float64
 5   mean compactness        569 non-null    float64
 6   mean concavity          569 non-null    float64
 7   mean concave points     569 non-null    float64
 8   mean symmetry           569 non-null    float64
 9   mean fractal dimension  569 non-null    float64
 10  radius error            569 non-null    float64
 11  texture error           569 non-null    float64
 12  perimeter error         569 non-null    float64
```

```
df.describe()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200

8 rows × 31 columns

SPLIT DATA

Data dibagi menjadi training set dan testing set dengan rasio 80:20 menggunakan train_test_split() agar model dapat diuji dengan data yang tidak dilatih sebelumnya. Selanjutnya, dilakukan standardisasi fitur menggunakan StandardScaler untuk meningkatkan kinerja SVM, sehingga model dapat bekerja lebih optimal dengan data yang memiliki skala yang seragam.

```
[ ]  # Membagi data menjadi train dan test
     X_train, X_test, y_train, y_test, = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[ ]  # Standardisasi fitur untuk meningkatkan kinerja SVM
     scaler = StandardScaler()
     X_train = scaler.fit_transform(X_train)
     X_test = scaler.transform(X_test)
```

TRAIN THE MODEL

Model SVM dibuat dengan menggunakan SVC() dari scikit-learn dan dilatih menggunakan training data untuk mempelajari pola klasifikasi antara tumor benign dan malignant.

```
[ ]  svm_model = SVC(kernel='rbf', C=1.0, gamma='scale')
     svm_model.fit(X_train, y_train)
```

```
SVC
SVC()
```

MODEL EVALUATION

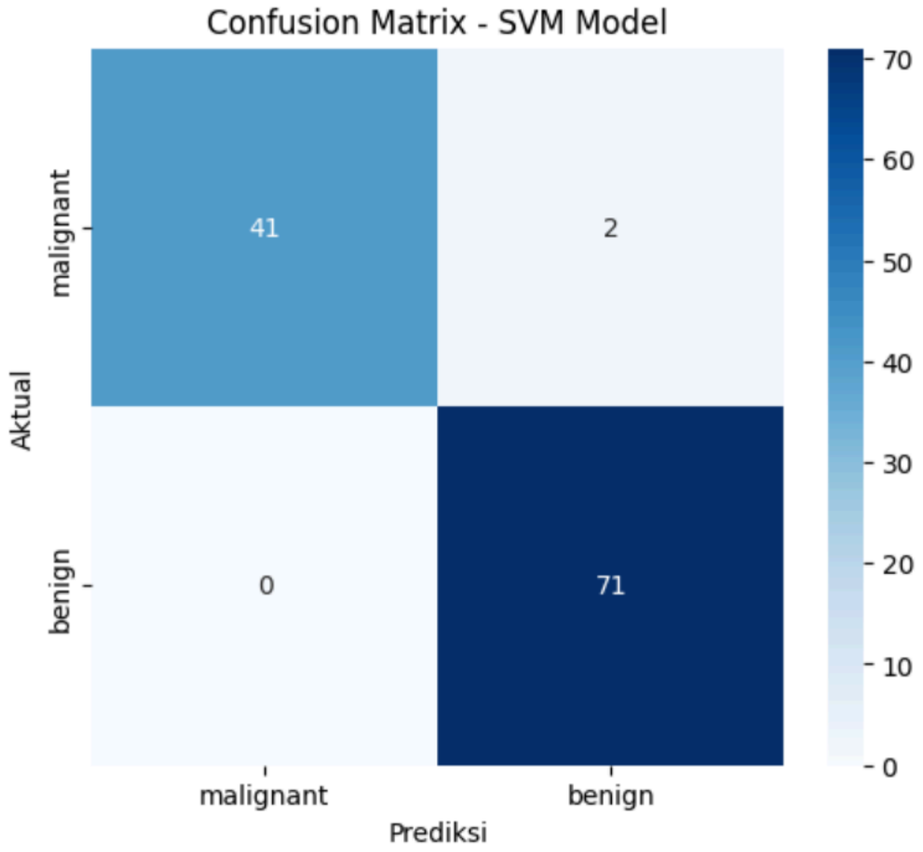
Model diuji menggunakan testing data, kemudian performanya diukur dengan accuracy score dan confusion matrix untuk melihat seberapa baik model dalam mengklasifikasikan data dengan benar. Dari hasil pengujian, model berhasil mencapai akurasi sebesar 98.25%, menunjukkan bahwa model mampu mengklasifikasikan data dengan tingkat ketepatan yang tinggi.

```
[ ]  # Prediksi data uji
     y_pred = svm_model.predict(X_test)

     # Evaluasi model
     accuracy = accuracy_score(y_test, y_pred)
     conf_matrix = confusion_matrix(y_test, y_pred)
     class_report = classification_report(y_test, y_pred)

     # Menampilkan hasil
     print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
Accuracy: 98.25%
```

VISUALIZATION

Confusion matrix menunjukkan bahwa model SVM memiliki akurasi 98.25%, dengan precision 100% untuk malignant dan recall 95.35%. Model berhasil mengklasifikasikan 41 sampel malignant (TP) dan 71 sampel benign (TN) dengan benar, namun masih terdapat 2 kasus False Negative (FN), yang berarti tumor ganas salah diklasifikasikan sebagai jinak. Untuk mengurangi False Negative, model dapat ditingkatkan dengan menyesuaikan threshold decision boundary, atau menerapkan balancing data jika distribusi kelas tidak seimbang.



THANK YOU!

081318956299

arjunamuhammadmalik@gmail.com

Arjuna Muhammad Malik