KOMPARASI METODE PREDIKSI RESTOCK DENGAN PENDEKATAN K-NEAREST NEIGHBOR (K-NN) DAN SUPPORT VECTOR MACHINE (SVM)

Eva Mahdyta Kiswana, Sukmadiningtyas
Sistem Informasi, Telkom University Purwokerto
Jl. DI Panjaitan No.128, Purwokerto Selatan, Indonesia
21103059@ittelkom-pwt.ac.id

ABSTRAK

Manajemen persediaan merupakan elemen penting dalam operasional supermarket untuk mencegah kelebihan atau kekurangan stok yang dapat menyebabkan kerugian finansial. Kesalahan dalam memprediksi dapat menyebabkan: kelebihan persediaan yang menyebabkan biaya penyimpanan yang tinggi dan potensi penurunan nilai barang, serta kekurangan persediaan yang dapat mengakibatkan hilangnya penjualan dan menurunnya kepuasan pelanggan. Penelitian ini bertujuan untuk membandingkan dua metode prediksi restock, yaitu *K-Nearest Neighbor* (K-NN) dan *Support Vector Machine* (SVM), dalam meningkatkan akurasi prediksi kebutuhan stok pada supermarket. Data penjualan digunakan sebagai basis untuk implementasi kedua algoritma tersebut. Tahapan penelitian meliputi pengumpulan data, preprocessing, implementasi algoritma, dan evaluasi kinerja menggunakan metrik akurasi. Hasil penelitian menunjukkan bahwa K-NN menghasilkan akurasi sebesar 88,75% dan SVM menghasilkan akurasi sebesar 88,26%, dengan K-NN memberikan performa lebih baik secara keseluruhan. Penelitian ini diharapkan dapat membantu supermarket dalam mengoptimalkan manajemen persediaan dan memberikan wawasan baru untuk pengembangan sistem prediksi stok yang lebih efektif.

Kata Kunci: *K-Nearest Neighbor (K-NN), Manajemen persediaan, Prediksi restock, Support Vector Machine (SVM), Supermarket.*

1. PENDAHULUAN

Saat ini, terdapat banyak sekali pertokoan di Indonesia khususnya di daerah Kabupaten Banyumas. Walaupun distribusi pertokoan sudah meluas, toko-toko masih sering menghadapi tantangan dalam mengelola persediaan produk secara efisien[1]. Supermarket menghadapi tantangan besar dalam manajemen persediaan, terutama dalam memprediksi permintaan produk dengan akurat dan menentukan jumlah *restock* yang optimal. Kesalahan dalam memprediksi dapat menyebabkan: kelebihan persediaan yang menyebabkan biaya penyimpanan yang tinggi dan potensi penurunan nilai barang, serta kekurangan persediaan yang dapat mengakibatkan hilangnya penjualan dan menurunnya kepuasan pelanggan[2].

Selain itu, komparasi prediksi *restock* dengan pendekatan *K-Nearest Neighbor* (K-NN) dan *Support Vector Machine* (SVM) pada supermarket bertujuan untuk mengevaluasi performa kedua metode ini dalam hal akurasi prediksi, efisiensi komputasi, dan kemampuan dalam menangani variasi data. Penelitian ini bertujuan untuk menentukan metode mana yang lebih baik dalam mengatasi masalah yang dihadapi pada supermarket, seperti fluktuasi musiman dalam permintaan, tren produk, dan kejadian tak terduga yang mempengaruhi permintaan. Adanya masalah ini belum ada metode yang tepat untuk menangani. Oleh karena itu, dibutuhkan metode peramalan atau *forecasting* yang dapat membantu untuk melakukan perkiraan *restock* barang.

Berdasarkan beberapa penelitian yang berkaitan tentang peramalan yaitu pemanfaatan metode perbandingan tiga algoritma *K-Nearest Neighbor* (K-NN), *Support Vector Machine* (SVM), dan Algoritma *Naïve Bayes* dalam memprediksi kebutuhan air minum dalam kemasan. Data yang digunakan untuk pengujian dari agen adimaru. Penelitian ini memanfaatkan metode klasifikasi untuk mengidentifikasi pola dalam data yang dapat digunakan untuk memprediksi kebutuhan *restock* dengan menggunakan ketiga algoritma tersebut[3].

Penelitian lain sebelumnya terdapat pemanfaatan metode yang lebih akurat dan efisien dalam memprediksi pergerakan harga saham di pasar. *Artificial Neural Networks* (ANN), dengan kemampuan adaptasinya yang tinggi terhadap data non-linear, dan *Support Vector Machines* (SVM), yang terkenal dengan efektivitasnya dalam klasifikasi dan regresi, kedua metode ini diuji menggunakan serangkaian data historis dari pasar saham[4].

Adanya persaingan yang ketat dan dinamika pasar yang cepat berubah, dalam mengelola persediaan, toko menggunakan berbagai strategi untuk memastikan ketersediaan barang yang tepat untuk memenuhi permintaan pelanggan tanpa menimbulkan kelebihan yang tidak perlu, toko membutuhkan pendekatan yang dapat membantu mengatasi permasalahan ini. Penelitian ini membandingkan efektivitas dua metode prediksi, yaitu *K-Nearest Neighbor* (K-NN) dan *Support Vector Machine* (SVM), untuk membantu supermarket meningkatkan akurasi prediksi permintaan pelanggan.

Penelitian dengan pendekatan algoritma *K-Nearest Neighbor* (K-NN) dan *Support Vector Machine* (SVM) telah dikenal sebagai dua metode yang efektif untuk memprediksi permintaan dan mengelola persediaan. Penelitian dilakukan dengan

menggunakan algoritma *K-Nearest Neighbor* (K-NN) dan *Support Vector Machine* (SVM) untuk memprediksi produk terlaris, membantu pemilik toko dalam merencanakan penyediaan stok produk dengan lebih mudah dan efisien[5]. Metode K-NN sangat efektif ketika training datanya yang sangat banyak. Dengan menggunakan *K-Nearest Neighbor* (K-NN) dapat membantu dalam memprediksi penjualan dan merencanakan penyediaan stok barang. Algoritma *K-Nearest Neighbor* (K-NN) adalah metode untuk mengklasifikasikan objek baru berdasarkan berdekatan dengan tetangganya yang berjumlah (K) [6]. *Support Vector Machine* (SVM) adalah algoritma yang memetakan vector input ke dalam ruang dimensi yang lebih tinggi dan menggunakan fungsi kernel untuk membangun hyperplane optimal yang memisahkan data[7]. Kedua metode tersebut berpotensi meningkatkan prediksi *restock* dan manajemen persediaan. Namun, penting untuk mempertimbangkan karakteristik data dan kebutuhan spesifik dari supermarket dalam memilih metode yang paling sesuai. Perbandingan langsung antara K-NN dan SVM diharapkan dapat mengungkap keunggulan dan kekurangan masing-masing metode dalam menyelesaikan masalah persediaan pada supermarket.

Penelitian ini bertujuan melakukan penentuan kebutuhan restock barang dengan memprediksi menggunakan metode K-NN dan SVM. Penentuan kedua metode ini berdasarkan tingkat efisiensi dan efektivitas. Oleh karena itu, penelitian ini mengambil judul komparasi metode prediksi *restock* dengan pendekatan *K-Nearest Neighbor* dan *Support Vector Machine* (SVM).

2. TINJAUAN PUSTAKA

2.1. K- Nearest Neighbor (K-NN)

K-Nearest Neighbor (K-NN) merupakan algoritma *supervised learning* yang mengklasifikasikan hasil dari *instance query* yang baru berdasarkan kategori mayoritas pada tetangga terdekat (K-NN. Algoritma *K-Nearest Neighbor* (K-NN) menggunakan klasifikasi tetangga terdekat sebagai nilai prediksi untuk *instance query* baru. Algoritma K-NN sangat sederhana dan beroperasi berdasarkan jarak terdekat dari *instance query* ke sampel. Proses pelatihan dilakukan dengan memproyeksikan data ke dalam tempat dengan banyak dimensi, di mana setiap dimensi mewakili fitur dari data tersebut. Ruang ini kemudian dipisahkan ke dalam beberapa bagian-bagian berdasarkan kategori sampel pelatihan. Suatu titik di ruang ini akan dikategorikan sebagai kelas c jika kelas c adalah klasifikasi yang paling sering muncul di antara k tetangga yang terdekat dengan titik itu. Jarak antar tetangga biasanya dihitung menggunakan jarak *Euclidean*. Cara-cara yang digunakan oleh algoritma *K-Nearest Neighbor* (K-NN)[8]:

a. Menentukan nilai K (jumlah tetangga terdekat).
b. Menghitung jarak kuadrat *Euclidean* antara *instance query* dan setiap objek dalam data sampel yang diberikan. Dengan rumus dibawah ini:

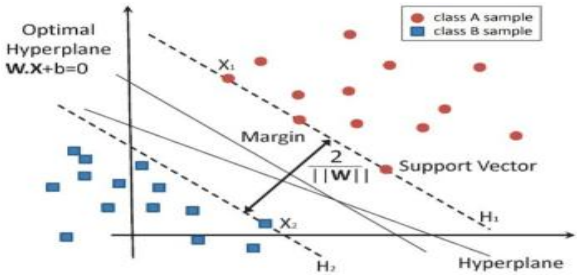
$$d_i = \sqrt{\sum_{t=1}^{n}(x_{2i} - x_{1i})^2} \quad (1)$$

Keterangan:
X_1 = Sampel data
X_2 = Data uji
i = Variabel data
d = Jarak
p = Dimensi data

c. Menyusun objek-objek tersebut ke dalam kategori dengan jarak Euclidean terkecil. Menggunakan kategori *nearest neighbor* yang paling dominan untuk memprediksi nilai query instance yang dihitung.

2.2. Support Vector Machine (SVM)

Prinsip dasar *Support Vector Machine (SVM)* adalah menemukan *hyperplane* (garis pemisah) yang dapat memisahkan dua kelas dengan optimal. Optimal dalam konteks ini berarti hyperplane tersebut mampu memisahkan data kedua kelas dengan margin terluas. *Margin* adalah seberapa jauh *hyperplane* dari titik-titik terdekat dari setiap kelompok. *Hyperplane* yang memisahkan kelas dengan margin terbesar disebut sebagai Optimal *Hyperplane*.



Gambar 1. Klasifikasi data menggunakan *Support Vector Machine* (SVM)

Gambar 1 menunjukkan bahwa garis H1, H2, dan *hyperplane* berfungsi sebagai pemisah antara dua kelas. Dalam hal ini, (X) merupakan hasil perkalian titik antara variabel dan konstanta pada setiap notasi, dan (W) adalah vektor yang tegak lurus terhadap (X).

$$w.Xi + b \leq -1 \quad (2)$$

Hyperplane ini adalah yang menyentuh data pada kelas A (H1).

$$w.Xi + b \geq +1 \quad (3)$$

Hyperplane ini adalah yang menyentuh data pada kelas B (H2).

$$w.X + b = 0 \quad (4)$$

Hyperplane berada di antara hyperplane untuk kelas A dan kelas B (Garis *Hyperplane*). Data yang menyentuh H1 di kelas A dan H3 di kelas B disebut *Support Vector*.

Proses pencarian titik minimal dikenal sebagai *Quadratic Programming (QP)*. Penentuan margin diperlukan untuk menentukan titik minimal, yang

dihitung dengan rumus $\frac{1}{||w||}$. Berikut adalah persamaan untuk menemukan titik minimal:

$$\min_w \tau\,(w) = \frac{1}{2}\,||w||^2 \qquad (5)$$

Dengan memperhatikan nilai *constrain*:

$$y_i(Xi.w + b) - 1 \geq 0,\ \forall_i \qquad (6)$$

Tidak semua sampel data dapat dipisahkan secara linear, sehingga penggunaan SVM linear tidak selalu memungkinkan. Memaksakan SVM linear pada data semacam itu akan menghasilkan klasifikasi yang kurang baik dan tidak efisien. Oleh karena itu untuk meningkatkan kinerja, SVM linear perlu diubah menjadi SVM non-linear yang menggunakan metode kernel. Ini melibatkan pendekatan yang berbeda dari metode klasifikasi biasa, di mana sering kali mengurangi dimensi awal untuk membuat proses komputasi lebih sederhana dan meningkatkan hasilnya[9].

2.3. Root Mean Square Error (RMSE)

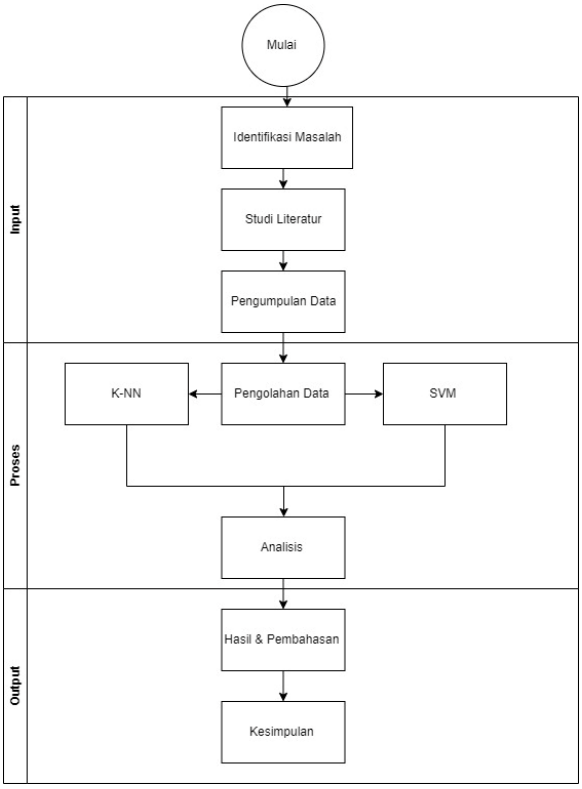
Root Mean Square Error (RMSE) digunakan untuk mengukur tingkat kesalahan dalam proses prediksi data. RMSE menghitung selisih antara nilai aktual dan nilai prediksi, kemudian membagi total jumlah kuadrat selisih tersebut dengan jumlah waktu prediksi, dan akhirnya mengambil akar kuadrat dari hasilnya[10].

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=0}^{n}(f_i + o_i)^2} \qquad (7)$$

Keterangan:
N = Nomor sample
F = Hasil peramalan
O = Nilai pengamatan

3. METODE PENELITIAN

Penelitian ini diawali dengan identifikasi masalah melalui observasi proses restock barang di supermarket. Dilanjutkan dengan studi literatur, yaitu pencarian jurnal relevan sejak tahun 2020 untuk memahami masalah dan solusi yang tepat. Pada tahap pengumpulan data**,** informasi penjualan seperti jenis produk, jumlah penjualan, dan waktu transaksi dikumpulkan untuk analisis. Data ini kemudian diolah menggunakan dua algoritma, yaitu *K-Nearest Neighbor* (K-NN) dan *Support Vector Machine* (SVM)**.** Proses pengolahan data dengan K-NN meliputi pengumpulan, normalisasi, pelatihan, dan pengujian data. Model dievaluasi berdasarkan akurasi klasifikasinya. Sementara itu, pengolahan data dengan SVM mencakup preprocessing, pelatihan, dan pengujian untuk menghasilkan hyperplane yang memisahkan data secara optimal. Pada tahap **analisis**, hasil dari kedua metode dibandingkan untuk menilai akurasi prediksi restock barang. Selanjutnya**,** hasil dan pembahasan membahas kelebihan masing-masing metode untuk menentukan mana yang lebih efektif. Akhirnya, penelitian ini ditutup dengan kesimpulan yang merangkum temuan dan rekomendasi untuk strategi restock yang optimal di supermarket.



Gambar 2. Diagram alir penelitian

4. HASIL DAN PEMBAHASAN

4.1. Pengumpulan Data

Data yang digunakan dalam penelitian ini adalah dataset penjualan supermarket yang diperoleh dari *platform*. Dataset ini mencakup data historis transaksi dari tiga cabang supermarket yang terletak di tiga kota yang berbeda dengan jumlah 1001 data. Berikut adalah sebagian data dari keseluruhan data yang digunakan.

Tabel 1. Dataset yang digunakan

No.	Product line	Unit price	Quantity	Total	Date
1.	Sports and travel	72.61	6	457.443	01/01/2019
2.	Home and lifestyle	47.59	8	399.756	01/01/2019
3.	Electronic accessories	74.71	6	470.673	01/01/2019
4.	Sports and travel	36.98	10	388.29	01/01/2019
5.	Electronic accessories	63.22	2	132.762	01/01/2019
6.	Health and beauty	62.87	2	132.027	01/01/2019
7.	Fashion accessories	65.74	9	621.243	01/01/2019
8.	Food and beverages	84.63	10	888.615	01/01/2019
9.	Sports and travel	27.04	4	113.568	01/01/2019
10.	Electronic accessories	74.22	10	779.31	01/01/2019

4.2. Implementasi Algoritma K-NN

4.2.1. Pengumpulan dan Penyiapan Data

Pada tahap ini, data penjualan diproses berdasarkan atribut seperti jenis produk, jumlah penjualan, dan waktu transaksi. Berikut lampiran *source code* tahapan dari pengumpulan dan penyiapan data.

a) Membuat dan membaca data

```
# Load data
data_path = '/content/drive/MyDrive/SMS. 7/Tugas Akhir/supermarket_sales - Sheet1.csv'
sales_data = pd.read_csv(data_path)
```

Pada lampiran *source code* diatas berfungsi untuk memuat data dari file CSV menggunakan pandas.read_csv.

b) Mengonversi kolom '*Date*' ke format *Datetime*

```
# Load data
data_path = '/content/drive/MyDrive/SMS. 7/Tugas Akhir/supermarket_sales - Sheet1.csv'
sales_data = pd.read_csv(data_path)
```

Pada lampiran *source code* diatas berfungsi untuk mengonversi kolom date menjadi format *datetime* untuk mempermudah menganalisis data berbasis waktu.

c) Ekstraksi tambahan dari kolom '*Date*'

```
# Extract year, month, and day for further analysis
sales_data['Year'] = sales_data['Date'].dt.year
sales_data['Month'] = sales_data['Date'].dt.month
sales_data['Day'] = sales_data['Date'].dt.day
sales_data['Weekday'] = sales_data['Date'].dt.weekday
```

Pada lampiran *source code* diatas berfungsi untuk menambahkan kolom baru untuk tahun, bulan, hari, dan hari dalam minggu, untuk menganalisis lebih rinci berdasarkan waktu.

d) Mengelompokkan data berdasarkan '*Product line*' dan '*Month*'

```
# Aggregate data by Product line and Month to summarize monthly quantities
monthly_sales = sales_data.groupby(['Product line', 'Month']).agg(
    Total_Quantity=('Quantity', 'sum'),
    Average_Unit_Price=('Unit price', 'mean')
).reset_index()
```

Pada lampiran *source code* diatas berfungsi untuk mengelompokkan data berdasarkan product line dan month, menghitung jumlah total kuantitas (Total_*Quantity*) perbulan, dan menghitung harga unit rata-rata (*Average_Unit_Price*) untuk setiap kategori.

e) Menyiapkan fitur dan target

```
# Prepare features and target variables
X = monthly_sales[['Month', 'Average_Unit_Price']]
y = monthly_sales['Total_Quantity']
```

Pada lampiran *source code* diatas menjelaskan :

- Variabel x : menentukan fitur (bulan dan harga rata-rata)
- Variabel y : menentukan target prediksi (jumlah kuantitas)

f) Standardisasi fitur

```
# Standardizing the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Pada lampiran *source code* diatas berfungsi melakukan standardisasi pada fitur untuk memastikan nilai-nilai berada pada skala yang sama, sehingga algoritma K-NN dapat bekerja lebih baik.

4.2.2. Normalisasi Data Numerik

Pada tahap ini data dinormalisasi untuk memastikan keseragaman skala antar atribut. Berikut lampiran *source code* tahapan dari normalisasi data numerik.

a) Standardisasi fitur

```
# Standardizing the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Pada lampiran *source code* diatas menjelaskan normalisasi bertujuan untuk memastikan fitur numerik memiliki skala yang sama. Penting untuk algoritma K-NN, karena algoritma ini menggunakan jarak *Euclidean* untuk mengukur kedekatan antara data poin.

b) Cara kerja *StandardScaler* :

1) *StandardScaler* dari *sklearn.preprocessing* mentransformasikan data sehingga memiliki mean = 0 dan standar deviasi = 1.
2) Rumus transformasi :

$$X_{scaled} = \frac{X-\mu}{\sigma} \quad (4.1)$$

Keterangan :
X : Nilai asli dari fitur
μ : Rata-rata dari fitur
σ : Standar deviasi dari fitur

3) Tahapan
fit_transform(X) :
Penjelasan :
   - fit : Menghitung rata-rata (μ) dan standar deviasi (σ) dari data.
   - transform : Mengaplikasikan rumus standardisasi ke setiap nilai dalam data.
4) Variabel hasil
X_scaled adalah data yang telah dinormalisasi, digunakan untuk proses pelatihan dan pengujian algoritma K-NN.

```
# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

Dengan langkah ini, semua fitur numerik (*Month* dan *Average_Unit_Price*) memiliki skala yang setara, sehingga algoritma K-NN dapat bekerja dengan lebih baik.

4.2.3. Pelatihan dan Pengujian Data

a) Membagi data menjadi *training* dan *testing* set

```
# Splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

Penjelasan :

- X_train : Data fitur untuk pelatihan algoritma.
- X_test : Data fitur untuk pengujian algoritma.
- y_train : Target variabel (jumlah total kuantitas) untuk pelatihan model.
- y_test : Target variabel untuk pengujian algoritma.
- test_size=0.2 : 20% data digunakan untuk pengujian, dan 80% data digunakan untuk pelatihan.
- random_state=42 : Menetapkan nilai random untuk memastikan hasil yang konsisten.

b) Melatih algoritma K-NN

```
# Initialize the KNeighborsRegressor
knn = KNeighborsRegressor(n_neighbors=5)

# Train the model
knn.fit(X_train, y_train)
```

Penjelasan :

- *KNeighborsRegressor*(n_neighbors=5) : Inisialisasi algoritma K-NN dengan parameter jumlah tetangga k = 5.
- fit(X_train, y_train) : Melatih algoritma K-NN menggunakan data pelatihan (x_train dan y_train).

c) Memprediksi data pengujian

```
# Predict on the testing set
y_pred = knn.predict(X_test)
```

Penjelasan :

- *predict*(X_*test*) : Menggunakan model yang telah dilatih untuk memprediksi target variabel pada data pengujian (X_*test*).
- y_pred : Hasil prediksi dari algoritma K-NN.

d) Menghitung *Mean Squared Error* (MSE)

```
# Calculate the Mean Squared Error (MSE) for model evaluation
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
```

Penjelasan :

- *mean_squared_error*(y_test, y_pred) : Mengukur seberapa jauh prediksi model menyimpang dari nilai sebenarnya.
- mse : Rata-rata dari kuadrat kesalahan prediksi (error).

e) Menghitung akurasi algoritma

```
# Define accuracy function based on total quantity
def accuracy(y_test, y_pred):
    return 1 - np.sqrt(np.mean((y_test - y_pred) ** 2)) / np.mean(y_test)

# Print accuracy
model_accuracy = accuracy(y_test, y_pred) * 100
print(f"Model Accuracy: {model_accuracy:.2f}%")
```

Penjelasan :

Fungsi *accuracy* digunakan untuk menghitung tingkat keakuratan algoritma berdasarkan rata-rata total kuantitas. Dan hasilnya dinyatakan dalam presentase.

4.2.4. Evaluasi Algoritma K-NN

Pada tahap ini, hasil evaluasi menunjukkan bagaimana algoritma *K-Nearest Neighbor* (K-NN) berforma berdasarkan data pengujian. Berikut adalah rincian evaluasi yang dihasilkan :

1) Nilai *Mean Squared Error* (MSE) memberikan gambaran tentang tingkat kesalahan kuadrat rata-rata antara nilai actual dan nilai prediksi. Nilai MSE yang kecil menunjukkan bahwa model memiliki kemampuan prediksi yang baik. Sebaliknya, nilai MSE yang besar mengindikasikan bahwa model kurang akurat dalam memprediksi data.
2) Akurasi model dihitung dengan membandingkan *Root Mean Squared Error* (RMSE) terhadap rata-rata nilai actual dalam data pengujian. Akurasi ini dinyatakan dalam persentase (%), yang memberikan indikasi seberapa baik model bekerja dalam skala yang mudah dipahami. Akurasi yang tinggi (mendekasi 100%) menunjukkan bahwa prediksi model hampir sama dengan data aktual.
3) Interpretasi hasil :
   a) Jika akurasi model > 90%, maka model dianggap sangat baik dalam memprediksi data.
   b) Jika akurasi model berada di kisaran 80%-90%, maka model dianggap baik namun masih memiliki ruang untuk perbaikan.
   c) Jika akurasi model < 80%, maka perlu dilakukan optimasi terhadap model, dengan menyesuaikan parameter jumlah tetangga (n_neighbors), normalisasi data, atau bahkan menggunakan teknik lain.

   Berikut untuk menampilkan hasil evaluasi.

```
print(f"Model Accuracy: {model_accuracy:.2f}%")
```

Hasil akurasi dari algoritma *K-Nearest Neighbor* (K-NN) adalah 88.75% .

4.3. Implementasi Algoritma Support Vector Machine (SVM)

4.3.1. Pengumpulan Data

Pada tahap ini, data penjualan diproses berdasarkan atribut seperti jenis produk, jumlah

penjualan, dan waktu transaksi. Berikut lampiran *source code* tahapan dari pengumpulan data.

a) Membuat dan membaca data

```
# Load the dataset
file_path = '/content/drive/MyDrive/SMS. 7/Tugas Akhir/supermarket_sales - Sheet1.csv'
data = pd.read_csv(file_path)
```

Pada lampiran *source code* diatas berfungsi untuk membuat data dari file CSV menggunakan pandas.read_csv.

4.3.2. Preprocessing Data

Pada tahap *preprocessing* data berikut langkah-langkah nya.

a) Konversi kolom tanggal

```
# Convert 'Date' to datetime format
data['Date'] = pd.to_datetime(data['Date'])
```

Pada lampiran *source code* diatas, untuk mengubah kolom *Date* menjadi format *datetime* untuk mempermudah analisis waktu.

b) Ekstraksi tahun dan bulan

```
# Convert 'Date' to datetime format
data['Date'] = pd.to_datetime(data['Date'])
```

Pada lampiran *source code* diatas, menambahkan kolom baru untuk *Year* dan *Month yang* diekstraksi dari kolom *Date*.

c) Pengelompokan Data

```
# Aggregate data by Product line and Month to summarize monthly quantities
monthly_sales = data.groupby(['Product line', 'Month']).agg(
    Total_Quantity=('Quantity', 'sum'),
    Average_Unit_Price=('Unit price', 'mean')
).reset_index()
```

Pada lampiran *source code* diatas, mengelompokkan data berdasarkan *Product line* dan *Month* untuk *menghitung* total kuantitas dan rata-rata harga unit.

d) Standarisasi Data

```
# Scale data using StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Pada lampiran *source code* diatas, menskalakan data numerik menggunakan *StandardScaler* agar fitur memiliki *distribusi* dengan rata-rata 0 dan standar deviasi 1.

4.3.3. Data Testing dan Data Training

Pada tahap ini, berikut *source code* data *testing* dan data *training*.

```
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

Penjelasan :

a) X_train : Data fitur untuk pelatihan algoritma.
b) X_test : Data fitur untuk pengujian algoritma.
c) y_train : Target variabel (jumlah total kuantitas) untuk pelatihan model.
d) y_test : Target variabel untuk pengujian algoritma.
e) test_size=0.2 : 20% data digunakan untuk pengujian, dan 80% data digunakan untuk pelatihan.
f) random_state=42 : Menetapkan nilai random untuk memastikan hasil yang konsisten.

4.3.4. Klasifikasi Algoritma SVM

a) Mengimpor SVR

```
from sklearn.svm import SVR
```

Pada lampiran *source code* diatas, merupakan model *Support Vector Regression*, yang digunakan untuk memprediksi nilai kontinu (jumlah kuantitas).

Membangun dan Melatih Model

```
# Train SVM model
svm_model = SVR()
svm_model.fit(X_train, y_train)
```

Penjelasan :

- SVR() membuat instance model SVR untuk regresi.
- Fit(x_train, y_train) melatih model SVM menggunakan data training (x_train) dan label target (y_train).

b) Melakukan Prediksi

```
svm_accuracy = accuracy(y_test, svm_model.predict(X_test))
print(f"SVM Accuracy: {svm_accuracy * 100:.2f}%")
```

Penjelasan :

- svm_model.predict(X_test) menghasilkan prediksi untuk data testing.
- Fungsi *accuracy* digunakan untuk menghitung akurasi model dengan membandingkan prediksi terhadap nilai actual (y_test).

c) Prediksi untuk bulan berikutnya

```
# Predict quantities for the next month
future_quantity_predictions = svm_model.predict(X_future_scaled)
```

Pada lampiran *source code* diatas, model SVM telah dilatih untuk memprediksi kuantitas bulan berikutnya berdasarkan data terbaru yang telah distandarisasi (X_future_scaled).

4.3.5. Pengujian Akurasi

a) Fungsi untuk menghitung akurasi

```
def accuracy(y_test, y_pred):
    return 1 - np.sqrt(np.mean((y_test - y_pred) ** 2)) / np.mean(y_test)
```

Penjelasan :

- Fungsi ini menghitung *Root Mean Squared Error* (RMSE) adalah rata-rata dari selisih kuadrat antara nilai aktual (y_test) dan nilai prediksi (y_pred), lalu diambil akar kuadratnya.
- Fungsi membandingkan RMSE dengan rata-rata nilai aktual (np.mean(y_test)) untuk memberikan rasio kesalahan.
- Dengan mengurangi rasio 1 fungsi mengembalikan tingkat keakuratan model.

b) Menguji akurasi dengan data testing

```
svm_accuracy = accuracy(y_test, svm_model.predict(X_test))
print(f"SVM Accuracy: {svm_accuracy * 100:.2f}%")
```

Penjelasan :

- svm_model.predict(X_test) model SVM membuat prediksi untuk data testing (X_test).
- accuracy(y_test, svm_model.predict(X_test)) fungsi accuracy membandingkan nilai aktual (y_test) dengan prediksi model (svm_model.predict(X_test).

Hasil akurasi dari algoritma *Support Vector Machine* (SVM) adalah 88.26%.

4.4. Perbandingan Tingkat Akurasi

Berikut adalah perbandingan akurasi dari kedua metode tersebut.

Tabel 2. Perbandingan tingkat akurasi

Metode	Akurasi	Penjelasan
K-NN	88.75%	Akurasi lebih tinggi, unggul sebesar 0,49% dibandingkan metode SVM. Hal ini menunjukkan bahwa K-NN lebih unggul dalam menangani dataset ini dibandingkan SVM.
SVM	88.26%	Akurasi sedikit lebih rendah dibandingkan K-NN. Namun, memiliki performa yang hampir setara, sehingga pemilihan metode dapat disesuaikan dengan kebutuhan spesifik, seperti kecepatan komputasi, kompleksitas model, atau interpretasi hasil.

5. KESIMPULAN DAN SARAN

Berdasarkan penelitian yang telah dilakukan, hasil implementasi menunjukkan bahwa metode *K-Nearest Neighbor* (K-NN) dan *Support Vector Machine* (SVM) efektif dalam memprediksi kebutuhan *restock* barang. K-NN memiliki keunggulan dalam mengklasifikasikan pola data penjualan historis dengan efisien, sedangkan SVM unggul dalam memisahkan kelas data yang memiliki tingkat kompleksitas lebih tinggi. Berdasarkan evaluasi kinerja dengan menggunakan metrik akurasi, K-NN mencapai akurasi 88,75%, yang sedikit lebih tinggi daripada SVM yang memperoleh 88,26%. Hal ini mengindikasikan bahwa dalam konteks penelitian ini, K-NN memiliki performa yang lebih baik, sehingga menjadi metode yang lebih sesuai untuk digunakan dalam prediksi kebutuhan *restock* barang. Berdasarkan temuan penelitian, terdapat rekomendasi dapat diajukan untuk pengembangan lebih lanjut, yaitu pemilihan parameter yang tepat, seperti jumlah tetangga (K) untuk K-NN atau jenis kernel untuk SVM, sangat berpengaruh terhadap hasil prediksi. Oleh karena itu, disarankan untuk menerapkan teknik optimasi parameter seperti grid search atau random search guna meningkatkan akurasi model.

DAFTAR PUSTAKA

[1] Y. Fabien, A. Saputra Kirsan, P. Sistem Informasi, J. Matematika dan Teknologi Informasi, and I. Teknologi Kalimantan, "Pengembangan Sistem Informasi Penjualan & Inventaris Toko (SIANTO) Berbasis Website," *SPECTA Journal of Technology*, vol. 6, no. 1, pp. 1–7, 2022, doi: 10.35718/specta.v6i1.

[2] M. Reza Pratama, A. Supriyanto, and J. Trilomba Juang No, "Sistem Prediksi Pemesanan Dan Pengendalian Stok Barang Menggunakan Metode EOQ Dan ROP Pada Apotek Setia Kawan Pati," *Jurnal Informatika & Rekayasa Elektronika)*, vol. 5, no. 1, pp. 1–11, 2022, [Online]. Available: http://e-journal.stmiklombok.ac.id/index.php/jireISSN.2620-6900

[3] R. Faujana, D. Pratiwi, S. Sumarlinda, and F. E. Nastiti, "International Journal of Information System Technology and Data Science (IJIST-DAS) Comparative Analysis of Restock Needs Bottled Water Using K-Nearest Neighbor (K-NN), Support Vector Machine (SVM), and the Naïve Bayes Algorithm," *International Journal of Information System Technology and Data Science (IJIST-DAS)*, vol. 1, no. 1, pp. 1–8, 2023, [Online]. Available: http://ejournal.lotusaruna.id/index.php/ijistdas

[4] A. Kurani, P. Doshi, A. Vakharia, and M. Shah, "A Comprehensive Comparative Study of Artificial Neural Network (ANN) and Support Vector Machines (SVM) on Stock Forecasting," *Annals of Data Science*, vol. 10, no. 1, pp. 1–26, Feb. 2023, doi: 10.1007/s40745-021-00344-x.

[5] I. Hardi Pratama and U. Salamah, "Perbandingan Algoritma K-NEAREST NEIGHBOR Dan SUPPORT VECTOR MACHINE Untuk Menentukan Prediksi Produk-produk Terlaris Pada Toko Madura Kecamatan Pondok Aren," *Jurnal Teknik Informatika Kaputama (JTIK)*, vol. 6, no. 2, pp. 1–13, 2022.

[6] A. Azlina Putri, "RESOLUSI : Rekayasa Teknik Informatika dan Informasi Penerapan Data Mining Untuk Memprediksi Penjualan Buah Dan Sayur Menggunakan Metode K-Nearest Neighbor (Studi Kasus : PT. Central Brastagi Utama)," *Media Online)*, vol. 1, no. 6, pp. 1–8,

2021, [Online]. Available: https://djournals.com/resolusi

[7] Y. M. Hutahaean and A. W. Wijayanto, "Klasifikasi Rumah Tangga Penerima Subsidi Listrik di Provinsi Gorontalo Tahun 2019 dengan Metode K-Nearest Neighbor dan Support Vector Machine," *Jurnal Sistem dan Teknologi Informasi (JustIN)*, vol. 10, no. 1, pp. 1–6, Jan. 2022, doi: 10.26418/justin.v10i1.51210.

[8] E. Qiudandra, R. Akram, and Novianda, "Sistem Pakar Diagnosa Penyakit Osteoarthritis Dengan Menggunakan Metode K-NEAREST NEIGHBOR," *Methotika : Jurnal Ilmiah Teknik Informatika*, vol. 2, no. 2, pp. 1–12, 2022, [Online]. Available: http://ojs.fikom-methodist.net/index.php/METHOTIKA

[9] M. D. Purbolaksono, M. Irvan Tantowi, A. Imam Hidayat, and A. Adiwijaya, "Perbandingan Support Vector Machine dan Modified Balanced Random Forest dalam Deteksi Pasien Penyakit Diabetes," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 5, no. 2, pp. 1–7, Apr. 2021, doi: 10.29207/resti.v5i2.3008.

[10] F. K. T. Putri and A. D. Wowor, "Implementasi Algoritma Long Short-Term Memory dalam Prediksi Konsentrasi Gas Metana (CH4) di Kota Salatiga," *Jurnal JTIK (Jurnal Teknologi Informasi dan Komunikasi)*, vol. 8, no. 2, pp. 1–9, 2024, doi: 10.35870/jti.