Analisis Spektrum Singular pada Peramalan Deret Waktu Multivariat

Muhammad Fathur Rizky - 13523105[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*rizkyfathur326@gmail.com*, *13523105@std.stei.itb.ac.id*

***Abstract*—This paper explores the application of Singular Spectrum Analysis (SSA) for forecasting food commodity prices in Bandung City using data from Indonesia's National Food Agency Panel (2021-2024). The study implements SSA through embedding, SVD decomposition, grouping, and diagonal averaging stages. Analysis reveals better performance on non-stationary data, with forecasting accuracy varying across commodities—from 6.36% MAPE for beef to 37.77% for red chili. A strong correlation between positive singular values and error metrics indicates that higher data variability leads to less accurate predictions. The results demonstrate SSA's effectiveness in analyzing food commodity price time series, particularly for data with simple patterns, while highlighting its ability to handle non-stationary data without additional transformations.**

***Keywords*—Multivariate Analysis, Time Series Forecasting, Singular Spectrum Analysis.**

I. Pendahuluan

Peramalan harga bahan pangan eceran memiliki peran yang sangat penting bagi pelaku usaha dalam menjaga stabilitas bisnis dan mendukung perencanaan strategis jangka panjang. Dengan memahami pola fluktuasi harga yang sering dipengaruhi oleh berbagai faktor, seperti musim, kondisi cuaca, tingkat permintaan pasar, distribusi, hingga kebijakan pemerintah, pelaku usaha dapat mengantisipasi risiko serta mengoptimalkan keuntungan. Informasi yang akurat tentang perubahan harga memungkinkan pelaku usaha, termasuk petani, pedagang, dan distributor, untuk mengambil keputusan yang lebih tepat, seperti menentukan waktu terbaik untuk membeli atau menjual produk, mengelola stok dengan lebih efisien, serta merancang strategi pemasaran yang optimal. Dengan demikian, kemampuan untuk memprediksi harga bahan pangan tidak hanya mendukung keberlanjutan bisnis, tetapi juga membantu meningkatkan daya saing di pasar.

Dalam beberapa dekade terakhir, berbagai metode telah dikembangkan untuk mendukung peramalan harga bahan pangan. Metode-metode ini mencakup pendekatan statistik tradisional, seperti ARIMA, hingga teknik berbasis pemelajaran mendalam. Salah satu pendekatan yang belakangan mendapatkan perhatian adalah Singular Spectrum Analysis (SSA). Metode ini dikenal karena kemampuannya dalam menangani data deret waktu yang kompleks, termasuk mengidentifikasi tren jangka panjang, pola musiman, dan gangguan atau noise dalam data. Keunggulan utama dari SSA adalah fleksibilitasnya karena metode ini tidak bergantung pada asumsi model tertentu. Hal ini membuat SSA sangat sesuai untuk menganalisis data harga bahan pangan, yang sering kali bersifat tidak teratur dan dinamis.

Penelitian ini bertujuan untuk mengimplementasikan SSA dalam memprediksi harga bahan pangan ke depan. Dengan fokus pada analisis dan peramalan, penelitian ini diharapkan dapat memberikan wawasan yang lebih mendalam tentang efektivitas SSA sebagai alat untuk memahami dan meramalkan pola harga bahan pangan, serta membantu pelaku usaha dalam mendukung pengambilan keputusan strategis berbasis data.

II. Dasar Teori

A. Singular Value Decomposition

Singular Value Decomposition adalah salah satu metode dekomposisi dalam aljabar linear yang digunakan untuk memfaktorkan sebuah matriks menjadi tiga matriks komponen.

$$A = U\sum V^T \tag{1}$$

Dengan A merupakan matriks yang ingin didekomposisi, U merupakan matriks ortogonal vektor singular kiri, $\sum$ merupakan matriks dengan nilai singular di diagonalnya, dan V^T merupakan transpose dari matriks ortogonal vektor singular kanan. Nilai singular σ adalah akar dari nilai eigen λ matriks AA^T atau A^TA [1].

B. Singular Spectrum Analysis

Singular Spectrum Analysis (SSA) adalah metode analisis data yang fleksibel dan kuat untuk menganalisis deret waktu. Metode ini menggabungkan elemen-elemen analisis deret waktu klasik, statistik multivariat, geometri multivariat, sistem dinamis, dan pengolahan sinyal. Tujuan utama SSA adalah mendekomposisi deret waktu menjadi komponen-komponen independen yang dapat diinterpretasikan, seperti tren, komponen periodik atau osilasi, dan noise [2]. Teknik ini banyak digunakan dalam berbagai bidang seperti matematika, fisika, ekonomi, meteorologi, dan penelitian pasar.

SSA terdiri dari empat tahap utama, yakni dekomposisi meliputi *embedding*, SVD, *grouping*, dan *diagonal averaging (hankelization)*. Tahap ini dijelaskan sebagai berikut.

1. *Embedding*
   Embedding mengubah deret waktu $Y_T = (y_1, y_2, \dots, y_T)$ menjadi matriks lintasan (*trajectory matrix*) X. Matriks ini dibuat dengan parameter *window length* L sehingga

$$X = \begin{bmatrix} y_1 & \cdots & y_K \\ \vdots & \ddots & \vdots \\ y_L & \cdots & y_T \end{bmatrix} \tag{2}$$

Dengan $K = T - L + 1$. Matriks X memiliki properti Hankel, yakni elemen sepanjang diagonal $i + j =$ konstanta memiliki nilai yang sama.

2. *Singular Value Decomposition*
   Pada tahap ini, dilakukan dekomposisi matriks *X* menjadi penjumlahan matriks dengan *rank* satu menggunakan SVD.
$$X = \sum_{i=1}^{d} \sqrt{\lambda_i}\, U_i V_i^T \quad (3)$$
   Dengan λ_i adalah nilai eigen dari XX^T. U_i adalah vektor eigen kiri, V_i adalah vektor utama (*principal components*) dan d adalah *rank* matriks X. Hasil dari langkah ini adalah koleksi tripel eigen $(\sqrt{\lambda_i}, U_i, V_i^T)$ yang dikenal sebagai eigentriples.
3. *Grouping*
   Pada langkah ini, eigentriples yang dihasilkan dari proses SVD dikelompokkan berdasarkan karakteristiknya (tren, osilasi, derau). Matriks hasil dari pengelompokkan dinyatakan sebagai berikut.
$$X = X_{I_1} + X_{I_2} + \cdots + X_{I_m} \quad (4)$$
   Dengan $I_1, I_2, \ldots, I_m$ adalah indeks kelompok.
4. *Diagonal Averaging (Hankelisasi)*
   Pada proses ini, dilakukan pengubahan matriks hasil pengelompokkan X_{I_K} kembali menjadi deret waktu $\tilde{Y}_T^{(k)}$ melalui proses hankelisasi. Hasil akhirnya adalah rekonstruksi deret waktu sebagai penjumlahan komponen.
$$y_t = \sum_{k=1}^{m} \tilde{y}_t^{(k)} \quad (5)$$

C. Recurrent Methodology pada Iterative Forecasting

Recurrent methodology memanfaatkan hubungan rekursif yang diturunkan dari basis vector singular untuk memprediksi nilai masa depan [3]. Hubungan rekursif ini secara matematis dinyatakan sebagai berikut.
$$x_{t+1} = \mathbf{R}^{\mathrm{T}} \cdot \mathbf{v}_{\mathrm{t}} \quad (6)$$
Dengan x_{t+1} merupakan nilai prediksi masa depan, $\mathbf{R}$ merupakan matriks prediksi rekursif $(L \times 1)$ yang diperoleh dari vektor singular, dan $\mathbf{v}_{\mathrm{t}}$ merupakan vektor nilai historis $(L -$ dimensi) yang diambil dari data terakhir $(x_t, x_{t-1}, \ldots, x_{t-L+1})$.

D. Uji Stasioneritas

Uji stasioneritas merupakan langkah awal yang krusial dalam analisis data deret waktu, terutama dalam penelitian ekonomi dan keuangan. Salah satu metode formal yang paling umum digunakan untuk menguji stasioneritas adalah Metode ADF. Metode ini adalah pengembangan dari Dickey-Fuller Test yang mengatasi masalah autokorelasi dalam error term dengan menambahkan lag variabel dependen (ΔY_t) ke dalam model [4].
$$\Delta Y_t = \alpha + \beta t + \gamma Y_{t-1} + \sum_{i=1}^{p} \delta_i \Delta Y_{t-1} + \epsilon_t \quad (7)$$
Dengan α adalah konstanta, βt adalah komponen tren, γY_{t-1} adalah koefisien *unit root*, ΔY_{t-1} adalah *lagged differencing* untuk mengeliminasi autokorelasi, dan *p* adalah jumlah lag.
Pengujian dilakukan dengan menggunakan nilai t-statistik dari koefisien γ yang dibandingkan dengan nilai kritis. Jika nilai t-statistik lebih kecil dari nilai kritis atau *p-value* < 0.05, H_0 ditolak dan data dianggap stasioner.

E. Metrik Penilaian

Dalam mengevaluasi performa model peramalan, pemilihan metrik penilaian yang tepat menjadi sangat penting untuk memastikan bahwa hasil peramalan sesuai dengan tujuan analisis. Dua metrik yang sering digunakan adalah *Mean Absolute Percentage Error* (MAPE) dan *Symmetric Mean Absolute Percentage Error* (sMAPE).

MAPE mengukur rata-rata kesalahan absolut dalam bentuk persentase terhadap nilai aktual. Rumus MAPE diberikan sebagai berikut.
$$\mathrm{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100 \quad (8)$$
Di mana y_i adalah nilai aktual, $\hat{y}_i$ adalah nilai prediksi, dan n adalah jumlah data. MAPE memberikan nilai yang mudah diinterpretasikan karena dinyatakan dalam bentuk persentase sehingga cocok untuk membandingkan performa model pada dataset dengan skala yang berbeda. Nilai MAPE yang lebih rendah menunjukkan model yang lebih akurat.

Sementara itu, sMAPE mengukur proporsi variansi dalam nilai aktual yang dapat dijelaskan oleh model peramalan. Rumus R^2 dirumuskan sebagai berikut.
$$\mathrm{sMAPE} = \frac{100}{n} \sum_{i=1}^{n} \frac{2 \cdot |y_{pred,i} - y_{true,i}|}{|y_{pred,i}| + |y_{true,i}|} \quad (9)$$

III. Implementasi

A. Penjelasan dan Penyiapan Dataset

Dataset yang dianalisis berasal dari Panel Harga Pangan Badan Pangan Nasional Indonesia, yang berisi informasi harga harian berbagai komoditas pangan di Kota Bandung. Dataset mencakup rentang waktu dari 1 Januari 2021 hingga 28 Desember 2024 dengan total 1459 kolom yang merepresentasikan tanggal, dan baris yang menggambarkan berbagai jenis komoditas. Kolom pertama, Komoditas (Rp), menyajikan nama-nama komoditas seperti Beras Premium, Beras Medium, Kedelai Biji Kering (Impor), Bawang Merah, Bawang Putih Bonggol, dan lain-lain. Data harga disajikan dalam satuan Rupiah (Rp).

Walaupun SSA tergolong metode analisis data deret waktu yang sangat fleksibel, tahapan SS—sebagaimana dijelaskan pada Bagian II-B—membuatnya lebih cocok untuk data nonstasioner. Dengan menggunakan Persamaan (7), didapatkan nilai hasil uji stasioner dengan menggunakan ADF sebagai berikut.

Tabel I
Uji Stasioner ADF

Kolom	t-stats	*p-value*	Stasioner?
Beras Premium	-0.936	0.776	N
Beras Medium	-1.307	0.626	N
Kedelai Biji Kering (Impor)	-2.166	0.219	N
Bawang Merah	-3.515	0.008	Y
Bawang Putih Bonggol	-1.111	0.711	N
Cabai Merah Keriting	-3.079	0.028	Y
Cabai Rawit Merah	-4.823	0	Y
Daging Sapi Murni	-1.625	0.47	N

Daging Ayam Ras	-5.816	0	Y
Telur Ayam Ras	-2.85	0.051	N
Gula Konsumsi	-1.056	0.732	N
Minyak Goreng Kemasan Sederhana	-3.578	0.006	Y
Tepung Terigu (Curah)	-2.172	0.216	N

Pada Tabel I, terdapat 4 kolom stasioner dan 9 kolom nonstasioner. Dengan mengetahui uji stasioner, diharapkan dapat meningkatkan interpretabilitas model dalam melakukan prediksi.

Dilakukan prapemrosesan untuk merapihkan dataset yang diberikan. Proses itu mencakup, pengubahan nilai '-' menjadi NaN. Selanjutnya, dilakukan pengubahan tipe data menjadi float agar kolom objek menjadi sebuah kolom numerik. Terakhir, dilakukan pengubahan tipe data tanggal yang semula objek menjadi DateTime yang kemudian dilakukan pengurutan agar menjadi sebuah deret waktu terurut.

B. Implementasi SSA

Pada tahap pertama, kelas SSA didefinisikan dengan variabel objek awal yang dirancang untuk menangani data deret waktu. Variabel-variabel ini mencakup data deret waktu itu sendiri, panjang deret waktu, dan frekuensi data. Metode `__init__` digunakan untuk menginisialisasi objek SSA. Dalam metode ini, data deret waktu disimpan dalam bentuk DataFrame dengan nama kolom yang diambil dari data input. Panjang data deret waktu *N* dihitung untuk menentukan ukuran embedding yang akan digunakan dalam proses embedding selanjutnya. Selain itu, frekuensi data (daily, monthly, yearly, dsb.) diidentifikasi secara otomatis melalui metode inferred_freq pada indeks DataFrame.

```
class SSA(object):
    def __init__(self, time_series):
        self.ts = pd.DataFrame(time_series)
        self.ts_name =
self.ts.columns.tolist()[0]
        if self.ts_name == 0:
            self.ts_name = 'ts'
        self.ts_v = self.ts.values
        self.ts_N = self.ts.shape[0]
        self.freq =
self.ts.index.inferred_freq
```

Fungsi embed dalam kelas Singular Spectrum Analysis (SSA) digunakan untuk melakukan proses embedding, yaitu mentransformasi data deret waktu menjadi sebuah matriks trajektori berdasarkan dimensi embedding tertentu. Fungsi ini bertujuan untuk merepresentasikan pola-pola dalam data deret waktu ke dalam bentuk matriks sehingga memungkinkan analisis lebih lanjut seperti dekomposisi dan rekonstruksi.

```
def embed(self, embedding_dimension=None,
suspected_frequency=None):
 if not embedding_dimension:
  self.embedding_dimension = self.ts_N//2
 else:
  self.embedding_dimension =
embedding_dimension
 if suspected_frequency:
  self.suspected_frequency =
suspected_frequency
  self.embedding_dimension = (
            self.embedding_dimension //
self.suspected_frequency)*self.suspected_freq
uency

 self.K= self.ts_N-self.embedding_dimension+1
 self.X = m(linalg.hankel(self.ts, np.zeros(
        self.embedding_dimension))).T[:,
:self.K]
 self.X_df = pd.DataFrame(self.X)
 self.X_complete = self.X_df.dropna(axis=1)
 self.X_com = m(self.X_complete.values)
 self.X_missing =
self.X_df.drop(self.X_complete.columns,
axis=1)
 self.X_miss = m(self.X_missing.values)
 self.trajectory_dimentions = self.X_df.shape
 self.complete_dimensions =
self.X_complete.shape
 self.missing_dimensions =
self.X_missing.shape
 self.no_missing = self.missing_dimensions[1]
== 0

 return self.X_df
```

Pada awalnya, fungsi ini menentukan dimensi embedding. Jika parameter `embedding_dimension` tidak diberikan, nilai defaultnya adalah setengah dari panjang data deret waktu. Selain itu, parameter opsional `suspected_frequency` memungkinkan penyesuaian dimensi embedding agar menjadi kelipatan dari frekuensi musiman tertentu yang mungkin terdapat dalam data. Hal ini berguna untuk menangkap pola musiman dengan lebih baik.

Setelah dimensi embedding ditentukan, fungsi ini membangun matriks trajektori menggunakan matriks Hankel sehingga elemen-elemen data deret waktu diatur ke dalam bentuk matriks persegi panjang. Matriks ini memiliki sejumlah baris sesuai dengan dimensi embedding dan sejumlah kolom *K*, yang dihitung berdasarkan panjang data dikurangi dimensi embedding ditambah satu. Matriks ini kemudian dipecah menjadi dua bagian: matriks lengkap yang tidak memiliki nilai hilang (`X_complete`) dan matriks yang memiliki nilai hilang (`X_missing`). Data lengkap dan data dengan nilai hilang juga diubah menjadi matriks matematis untuk mempermudah analisis berikutnya.

Fungsi `embed` juga menghitung dimensi dari matriks trajektori, baik untuk keseluruhan data, data yang lengkap, maupun data yang memiliki nilai hilang. Informasi ini digunakan untuk menentukan apakah matriks memiliki nilai hilang dengan mengatur variabel `no_missing`.

```
def decompose(self):
 X = self.X_com
 self.S = X*X.T
 self.U, self.s, self.V = linalg.svd(self.S)
 self.U, self.s, self.V = m(self.U),
np.sqrt(self.s), m(self.V)
 self.d = np.linalg.matrix_rank(X)
 Vs, Xs, Ys, Zs = {}, {}, {}, {}
 for i in range(self.d):
  Zs[i] = self.s[i]*self.V[:, i]
  Vs[i] = X.T*(self.U[:, i]/self.s[i])
  Ys[i] = self.s[i]*self.U[:, i]
  Xs[i] = Ys[i]*(m(Vs[i]).T)
 self.Vs, self.Xs = Vs, Xs
```

```
 self.s_contributions =
self.get_contributions(X, self.s, False)
 self.r =
len(self.s_contributions[self.s_contributions
> 0])
 self.r_characteristic = round(
(self.s[:self.r]**2).sum()/(self.s**2).sum(),
4)
 self.orthonormal_base = {i: self.U[:, i] for
i in range(self.r)}
```

Fungsi decompose dalam kelas Singular Spectrum Analysis atau SSA bertujuan untuk melakukan dekomposisi matriks trajektori dengan menggunakan metode Singular Value Decomposition atau SVD. Dekomposisi ini memecah matriks trajektori menjadi tiga komponen utama, yaitu singular values (`s`), singular left vectors (`U`), dan singular right vectors (`V`), yang merepresentasikan pola-pola dominan dalam data deret waktu sebagaimana pada Persamaan (3). Langkah pertama dalam fungsi ini adalah menghitung matriks kovarian (`S`), yang merupakan hasil perkalian antara matriks trajektori (`X_com`) dan transposenya (`X_com.T`). Matriks kovarian ini kemudian didekomposisi menggunakan SVD untuk menghasilkan tiga komponen utama, yaitu `U`, `s`, dan `V`.

Nilai singular (`s`) menunjukkan kontribusi setiap pola terhadap total variasi data. Vektor singular kiri (`U`) menggambarkan pola utama dalam data, sementara vektor singular kanan (`V`) menunjukkan hubungan antara pola tersebut dengan data asli. Setelah dekomposisi selesai, fungsi ini juga menghitung *rank* dari matriks trajektori yang disimpan dalam variabel `d`, yaitu jumlah singular values yang signifikan berdasarkan kontribusinya terhadap total variasi data. Selain itu, fungsi membangun beberapa komponen tambahan, seperti `Zs`, yaitu singular values yang dikalikan dengan kolom-kolom matriks `V`, `Vs`, yaitu vektor kanan (`V`) yang diskalakan, `Ys`, yaitu singular values yang dikalikan dengan kolom-kolom matriks `U`, dan `Xs`, yaitu matriks hasil rekonstruksi untuk masing-masing pola.

```
def get_contributions(X=None, s=None):
 lambdas = np.power(s, 2)
 frob_norm = np.linalg.norm(X)
 ret = pd.DataFrame(lambdas/(frob_norm**2),
columns=['Contribution'])
 ret['Contribution'] =
ret.Contribution.round(4)
 return ret[ret.Contribution > 0]
```

Fungsi ini juga menghitung kontribusi relatif dari setiap singular value terhadap total variasi data menggunakan metode get_contributions, dan hasilnya disimpan dalam variabel `s_contributions` yang memenuhi perumusan sebagai berikut. Pertama-tama, nilai singular dan nilai eigen memiliki hubungan sebagaimana pada Persamaan (10).

$$\lambda_i = s_i^2 \tag{10}$$

Dengan λ_i merepresentasikan variansi yang dibaca oleh nilai singular ke-i. Setelah mendapatkan nilai singular, dapat dihitung Frobenius Norm untuk menghitung total variansi pada matriks sebagaimana pada Persamaan (11).

$$\|X\|_F^2 = \sum_{i=1}^{n} \lambda_i \tag{11}$$

Persamaan (10) dan (11) digunakan untuk menghitung kontribusi relatif untuk setiap nlai singular sebagaimana pada Persamaan (12).

$$\text{Contribution}_i = \frac{s_i^2}{\|X\|_F^2} = \frac{s_i^2}{\sum_{j=1}^{n} s_j^2} \tag{12}$$

Berdasarkan hasil kontribusi ini, fungsi menentukan jumlah komponen signifikan yang disimpan dalam variabel `r`, yang merupakan nilai singular dengan kontribusi lebih besar dari nol. Fungsi juga menghitung karakteristik proyeksi, yaitu proporsi variasi yang dijelaskan oleh komponen signifikan tersebut, dan menyimpannya dalam variabel `r_characteristic`. Selanjutnya, fungsi ini membangun basis ortonormal dari vektor singular kiri (`U`) untuk komponen signifikan, yang disimpan dalam variabel `orthonormal_base`. Basis ini akan digunakan dalam proses rekonstruksi dan peramalan.

Secara keseluruhan, fungsi decompose bertujuan untuk mengekstraksi pola-pola dominan dari data deret waktu dan menyediakan struktur data dalam bentuk `Vs`, `Xs`, dan `orthonormal_base` yang dapat digunakan untuk langkah-langkah berikutnya, seperti rekonstruksi atau peramalan.

```
def _forecast_prep(self,
singular_values=None):
 self.X_com_hat =
np.zeros(self.complete_dimensions)
 self.verticality_coefficient = 0
 self.forecast_orthonormal_base = {}
 if singular_values:
  try:
   for i in singular_values:
    self.forecast_orthonormal_base[i] =
self.orthonormal_base[i]
  except:
   if singular_values == 0:
    self.forecast_orthonormal_base[0] =
self.orthonormal_base[0]
   else:
    raise ('Error while preparing')
 else:
  self.forecast_orthonormal_base =
self.orthonormal_base
  self.R =
np.zeros(self.forecast_orthonormal_base[0].sh
ape)[:-1]
  for Pi in
self.forecast_orthonormal_base.values():
   self.X_com_hat += Pi*Pi.T*self.X_com
   pi = np.ravel(Pi)[-1]
   self.verticality_coefficient += pi**2
   self.R += pi*Pi[:-1]
 self.R = m(self.R/(1-
self.verticality_coefficient))
 self.X_com_tilde =
self.diagonal_averaging(self.X_com_hat)
```

Fungsi `_forecast_prep` dalam kelas Singular Spectrum Analysis atau SSA bertujuan untuk mempersiapkan komponen-komponen yang dibutuhkan dalam proses peramalan berbasis rekursif. Langkah pertama dalam fungsi ini adalah menginisialisasi matriks `X_com_hat` sebagai matriks nol dengan dimensi yang sama seperti matriks lengkap atau `X_complete`. Selain itu, variabel `verticality_coefficient` juga diinisialisasi dengan nilai nol, yang nantinya akan digunakan untuk menghitung koefisien vertikalitas pada data. Fungsi ini juga mendefinisikan dictionary

`forecast_orthonormal_base`, yang digunakan untuk menyimpan basis ortonormal dari vektor singular yang relevan.

Jika parameter `singular_values` diberikan, fungsi akan memilih vektor singular yang relevan berdasarkan indeks yang diberikan dalam `singular_values`. Vektor singular yang terpilih akan disimpan dalam dictionary `forecast_orthonormal_base`. Jika parameter `singular_values` tidak diberikan, semua singular vectors dari `orthonormal_base` akan digunakan untuk peramalan. Setelah itu, fungsi menginisialisasi matriks **R** sebagai matriks nol dengan panjang yang satu elemen lebih sedikit dari dimensi vektor singular. Matriks **R** nantinya akan digunakan untuk menghitung proyeksi data dalam peramalan.

Selanjutnya, fungsi melakukan iterasi melalui setiap vektor singular dalam `forecast_orthonormal_base`. Pada setiap iterasi, vektor singular yang dipilih dikalikan dengan transposenya, dan hasilnya digunakan untuk memperbarui matriks `X_com_hat` dengan menambahkan kontribusinya ke matriks tersebut. Elemen terakhir dari vektor singular tersebut juga digunakan untuk memperbarui nilai `verticality_coefficient`, yang merepresentasikan kontribusi total dari vektor singular terhadap data. Elemen-elemen ini juga digunakan untuk memperbarui matriks **R**, yang menyimpan informasi proyeksi dari nilai historis ke nilai masa depan.

Setelah iterasi selesai, matriks **R** disesuaikan dengan membagi elemen-elemennya dengan satu dikurangi nilai `verticality_coefficient`. Proses ini memastikan bahwa proyeksi dilakukan secara akurat dengan mempertimbangkan kontribusi dari vektor singular. Terakhir, matriks `X_com_hat` yang telah diperbarui digunakan untuk melakukan proses diagonal averaging, menghasilkan `X_com_tilde` sebagai bentuk rekonstruksi akhir dari data. Secara keseluruhan, fungsi ini mempersiapkan semua komponen yang diperlukan untuk peramalan dengan menggunakan vektor singular dan matriks proyeksi.

C. Interpretabilitas Model

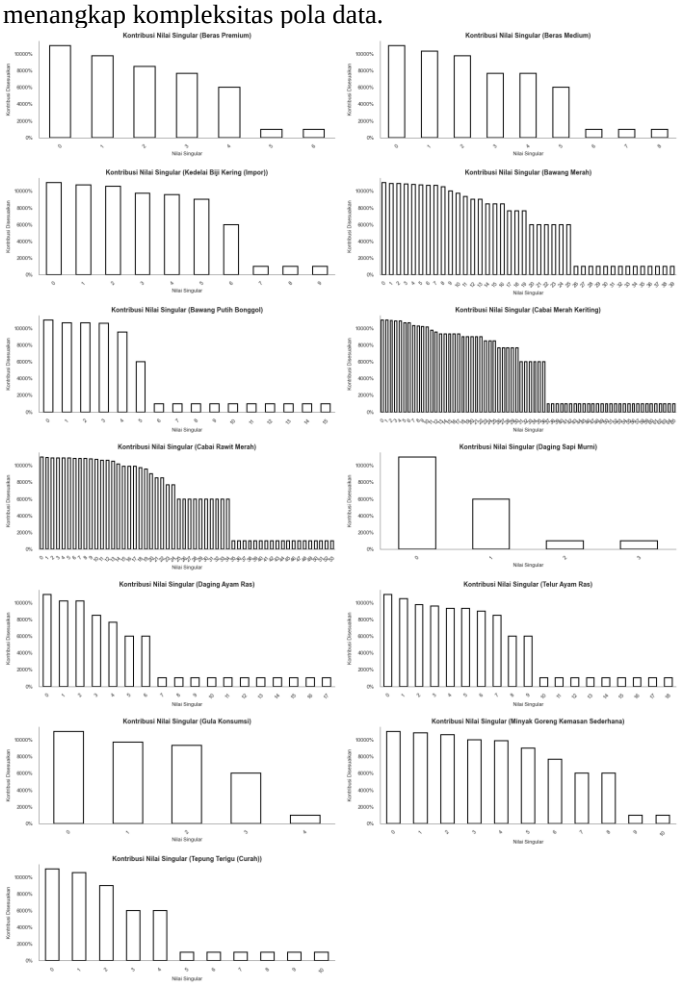
Interpretabilitas memungkinkan pengguna untuk memahami bagaimana sebuah model menghasilkan prediksi atau kesimpulan berdasarkan data input. Hal ini sangat relevan untuk memberikan kepercayaan terhadap hasil model serta membantu dalam pengambilan keputusan yang berbasis data. Dalam analisis deret waktu, interpretabilitas sering kali dihubungkan dengan bagaimana kontribusi variabel atau komponen tertentu dapat memengaruhi hasil model.

Pada model SSA, dapat dihitung kontribusi setiap nilai singular. Perhitungan ini bertujuan dalam melakukan *grouping* setiap sinyal yang berkontribusi. Perhitungan nilai kontribusi dilakukan berdasarkan Persamaan (12). Setelah dilakukan perhitungan, dilakukan penyesuaian skala dengan formula pada Persamaan (13).

$$\text{Contribution}_i = \max\left(\frac{1}{\text{Contribution}}\right) \cdot 1.1 - \frac{1}{\text{Contribution}_i} \quad (13)$$

Dengan menggunakan Persamaan (13), didapat kontribusi sebagaimana pada Gambar 1. Terlihat bahwa nilai singular pertama memiliki kontribusi terbesar di sebagian besar kolom data, yang menunjukkan bahwa pola utama atau tren global dalam data didominasi oleh komponen pertama. Namun, pada beberapa kolom seperti "Bawang Merah" dan "Cabai Merah Keriting," distribusi kontribusi lebih tersebar di antara banyak nilai singular, menandakan bahwa data pada kolom tersebut memiliki pola yang lebih kompleks atau variabilitas yang tinggi. Sebaliknya, pada kolom seperti "Gula Konsumsi" dan "Daging Sapi Murni," kontribusi didominasi oleh satu atau dua nilai singular pertama, yang mengindikasikan pola data yang lebih sederhana dan dapat direpresentasikan secara efektif dengan sedikit komponen utama.

Dengan memahami kontribusi sebagaimana ditampilkan pada Gambar 1, interpretabilitas model dapat ditingkatkan. Komponen dengan kontribusi besar dapat diidentifikasi sebagai elemen penting yang merepresentasikan pola utama dalam data, seperti tren global atau musiman, sementara komponen dengan kontribusi kecil cenderung mencerminkan *noise* atau variabilitas acak. Dengan mengabaikan komponen dengan kontribusi kecil, kita dapat menyederhanakan model tanpa kehilangan informasi penting, seperti yang terlihat pada kolom dengan distribusi kontribusi tajam seperti "Gula Konsumsi." Sebaliknya, kolom dengan kontribusi yang tersebar, seperti "Cabai Merah Keriting," memerlukan lebih banyak komponen untuk menangkap kompleksitas pola data.



Gambar 1. Kontribusi setiap nilai singular untuk setiap kolom

D. Prediksi dan Evaluasi

Fungsi forecast_recurrent dalam SSA digunakan untuk melakukan peramalan data deret waktu secara rekursif untuk

sejumlah langkah ke depan. Proses dimulai dengan memeriksa apakah komponen `X_com_hat` telah dihitung sebelumnya. Jika belum, fungsi `_forecast_prep` akan dipanggil untuk mempersiapkan matriks dan parameter yang diperlukan, termasuk matriks proyeksi **R**. Setelah itu, fungsi menginisialisasi variabel `ts_forecast` sebagai array yang berisi nilai awal data deret waktu, yaitu elemen pertama dari `ts_v`.

Proses peramalan dilakukan secara iteratif untuk setiap langkah ke depan sebagaimana pada Persamaan (6) sehingga mencapai jumlah langkah yang ditentukan oleh parameter `steps_ahead`. Dalam setiap iterasi, fungsi memeriksa apakah nilai aktual dari data deret waktu pada indeks tertentu adalah NaN atau tidak. Jika nilai tersebut adalah NaN, fungsi menggunakan matriks proyeksi **R** untuk menghitung nilai perkiraan. Proses ini melibatkan pengambilan nilai historis dari `ts_forecast` dalam rentang tertentu, kemudian mengalikannya dengan transpos dari matriks **R**. Hasil perkalian tersebut digunakan sebagai nilai perkiraan untuk langkah tersebut, yang kemudian ditambahkan ke array `ts_forecast`.

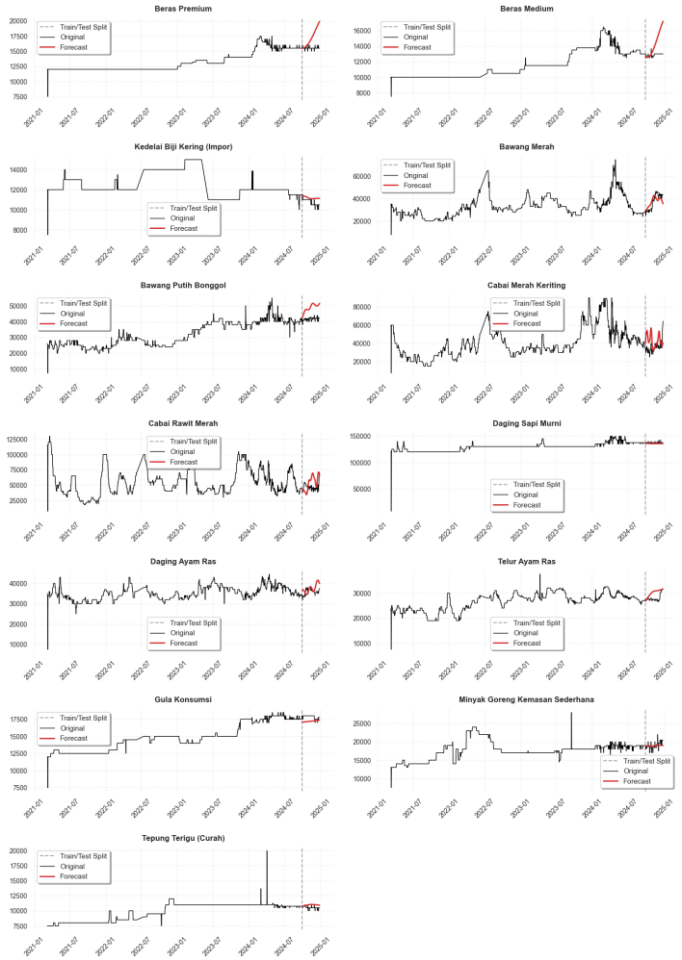
Jika nilai aktual data deret waktu pada indeks tertentu bukan NaN, nilai tersebut langsung ditambahkan ke `ts_forecast` sebagai bagian dari data historis. Apabila indeks iterasi melebihi panjang data deret waktu asli, fungsi akan menggunakan nilai historis terakhir yang tersedia dalam `ts_forecast` untuk melakukan peramalan. Proses ini dilakukan dengan cara yang sama, yaitu dengan mengalikan nilai historis dengan transpos dari matriks **R** dan menambahkan hasilnya ke `ts_forecast`.

Setelah iterasi selesai, fungsi menghitung jumlah total langkah peramalan, yang disimpan dalam variabel `forecast_N`. Fungsi kemudian membuat indeks waktu baru dengan memulai dari indeks waktu minimum data asli dan memperpanjangnya sesuai dengan jumlah langkah peramalan.

```
def forecast_recurrent(self, steps_ahead=12,
singular_values=None):
 try:
  self.X_com_hat
 except AttributeError:
  self._forecast_prep(singular_values)
 self.ts_forecast = np.array(self.ts_v[0])
 for i in range(1, self.ts_N+steps_ahead):
  try:
   if np.isnan(self.ts_v[i]):
    x = self.R.T * \
    m(self.ts_forecast[max(0, i-
self.R.shape[0]): i]).T
    self.ts_forecast =
np.append(self.ts_forecast, x[0])
   else:
    self.ts_forecast = np.append(
             self.ts_forecast, self.ts_v[i])
  except IndexError:
   x = self.R.T*m(self.ts_forecast[i-
self.R.shape[0]: i]).T
   self.ts_forecast =
np.append(self.ts_forecast, x[0])
  self.forecast_N = i+1
  new_index = pd.date_range(
  start=self.ts.index.min(),
periods=self.forecast_N, freq=self.freq)
  forecast_df =
pd.DataFrame(self.ts_forecast, columns=[
                'Forecast'],
index=new_index)
  forecast_df['Original'] =
np.append(self.ts_v, [np.nan]*steps_ahead)
  return forecast_df
```

Terdapat tiga tahap utama dalam melakukan prediksi, yakni pemanggilan metode `embed` untuk melakukan pengubahan deret waktu sebagaimana pada Persamaan (2), `decompose` untuk melakukan dekomposisi menggunakan SVD sebagaimana pada Persamaan (3), dan `forecast_recurrent` untuk melakukan peramalan sebanyak `steps_ahead` sebagaimana pada Persamaan (6).

Setelah dilakukan tiga tahap sebelumnya $\text{embedding\_dimension} = \left\lfloor \frac{N}{2} \right\rfloor$ dan $\text{suspected\_frequency} = 180$, dilakukan plot grafik hasil forecasting vs. original seperti pada Gambar 2. Terlihat bahwa terdapat hasil yang mendekati data orisinal, tetapi ada juga yang menyimpang dari seharusnya. Untuk lebih memudahkan dalam mengecek seberapa besar kesalahan peramalan, maka dilampirkan pula hasil galat forecasting pada Tabel II.



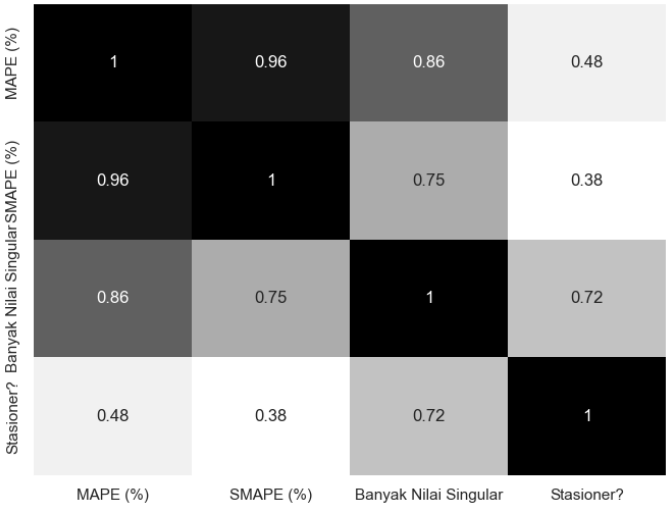
Gambar 2. Grafik peramalan vs. orisinal setiap kolom

TABEL II
SKOR PERAMALAN BASELINE

Komoditas	Metrik		Banyak Singularitas
	MAPE	sMAPE	
Beras Premium	15.75	17.35	6

Beras Medium	14.85	15.97	9
Kedelai Biji Kering (Impor)	15.73	14.11	10
Bawang Merah	26.7	28.38	40
Bawang Putih Bonggol	24.34	29.28	15
Cabai Merah Keriting	36.02	31.31	66
Cabai Rawit Merah	37.77	32.11	54
Daging Sapi Murni	6.36	6.7	4
Daging Ayam Ras	7.86	7.99	19
Telur Ayam Ras	10.59	11.29	19
Gula Konsumsi	16.23	18.36	5
Minyak Goreng Kemasan Sederhana	11.52	12.36	11
Tepung Terigu (Curah)	11.17	12.17	11
Rerata	18.07	18.26	20.69

Pada Tabel II, terlihat bahwa MAPE dan sMAPE yang dihasilkan cukup varians. Karena itu, dilakukan perbandingan dengan banyaknya nilai singular yang positif untuk mengetahui keterhubungannya. Terlihat pada Gambar 3, banyak nilai singular memiliki hubungan yang kuat dengan nilai galat MAPE (0.86) dan sMAPE (0.75).



Gambar 3. Matriks korelasi metrik dan banyak nilai singular

Dari Gambar 3, dapat diambil sedikit ringkasan bahwa semakin banyak nilai singular positif yang terdefinisi, semakin tinggi tingkat variabilitas kolom tersebut. Selain itu, kestasioneran memiliki korelasi positif dengan hasil galat MAPE dan sMAPE. Hal ini mendukung hipotesis awal sebagaimana pada Bagian III-A, bahwa kecocokan SSA cenderung pada data nonstasioner.

IV. Kesimpulan

Berdasarkan hasil analisis yang telah dilakukan, Singular Spectrum Analysis (SSA) terbukti sebagai metode yang efektif untuk menganalisis deret waktu pada data harga komoditas pangan. Metode ini mampu mengidentifikasi pola utama seperti tren, musiman, dan noise, sehingga memberikan pemahaman yang lebih mendalam terhadap struktur data. SSA juga memungkinkan reduksi dimensi data dengan memilih komponen nilai singular yang signifikan, yang pada akhirnya meningkatkan interpretabilitas model.

Pada data yang dianalisis, SSA menunjukkan kemampuan yang baik dalam merepresentasikan pola data sederhana dengan hanya menggunakan sedikit komponen utama, sementara data dengan pola kompleks memerlukan lebih banyak komponen untuk mencapai hasil yang optimal. Pengujian stasioneritas menunjukkan bahwa SSA mampu bekerja dengan baik pada data nonstasioner tanpa memerlukan transformasi tambahan sehingga meningkatkan fleksibilitas metode ini dalam berbagai jenis data deret waktu.

V. Saran Pengembangan

SSA memiliki parameter penting, seperti *embedding dimension* atau *window length L* yang digunakan untuk menentukan seberapa besar interval pertimbangan dalam melakukan peramalan ke depan. Selain itu, terdapat pula *suspected frequency* yang digunakan untuk mengasumsikan keperiodikan suatu deret waktu yang dapat membantu model untuk lebih melakukan *fitting* terhadap data dengan mempertimbangkan pola kualitatif. Kedua parameter tersebut dapat dilakukan *hyperparameter tuning* guna meminimalisasi nilai galat yang timbul.

VI. Lampiran

Github: https://github.com/fathurwithyou/Algeo-Forecasting

VII. Penutup

Ucapan terima kasih penulis berikan kepada Tuhan Yang Maha Esa, Institut Teknologi Bandung, Prodi Teknik Informatika, dan dosen mata kuliah IF2123 Aljabar Linier dan Geometri, Bapak Rila Mandala. Diharapkan makalah ini dapat bermanfaat bagi pembaca dan dapat dikembangkan lebih lanjut. Seperti kata Mushashi dalam blog Rinaldi Munir, "*Hidup lebih dari harus sekedar tetap hidup. Masalahnya, bagaimana membuat hidup itu bermakna, memancarkan cahaya gemilang ke masa depan, sekalipun harus mengorbankan hidup itu sendiri. Bila ini sudah tercapai, maka tidak ada bedanya dengan berapa lama hidup itu.*"

Referensi

[1] M. E. Wall, A. Rechtsteiner, and L. M. Rocha, "Singular Value Decomposition and Principal Component Analysis," in *A Practical Approach to Microarray Data Analysis*, D. P. Berrar, W. Dubitzky, and M. Granzow, Eds., Boston, MA: Springer US, 2003, pp. 91–109. doi: 10.1007/0-306-47815-3_5.

[2] H. Hassani, "Munich Personal RePEc Archive Singular Spectrum Analysis: Methodology and Comparison Singular Spectrum Analysis: Methodology and Comparison," 2007.

[3] N. Golyandina, V. Nekrutkin, and A. A. Zhigljavsky, "Analysis of Time Series Structure - SSA and Related Techniques," in *Monographs on statistics and applied probability*, 2001. [Online]. Available: https://api.semanticscholar.org/CorpusID:198120375

[4] R. Mushtaq, "Augmented Dickey Fuller Test," 2011, doi: 10.2139/ssrn.1911068.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 1 Januari 2025

Muhammad Fathur Rizky