Jurnal Penelitian
Ilmu Humaniora

JPIH, 7 (11), November 2024
ISSN: 21155640

PENAKSIRAN PARAMETER COX PROPORTIONAL HAZARD REGRESSION PADA DATA BESAR MENGGUNAKAN STOCHASTIC GRADIENT DESCENT

Dion Orlando Sitohang[1], Sutarman[2]

Abstrak: Penaksiran parameter model regresi Cox Proportional Hazard (CoxPH) sering menghadapi tantangan pada dataset besar. Dalam penelitian ini, metode Newton-Raphson dibandingkan dengan metode Stochastic Gradient Descent (SGD) untuk mengevaluasi penaksiran parameter. Log-partial likelihood dimanfaatkan untuk menaksir parameter model, dan dievaluasi menggunakan nilai Concordance Index (C-Index) sebagai metrik utama. Hasil menunjukkan bahwa SGD lebih unggul dalam semua ukuran dataset yang diuji. Pada dataset berukuran 10.000 sampel, SGD mencapai C-Index 0,683, sementara Newton-Raphson hanya 0,674. Selain itu, pada dataset berukuran 50.000 dan 100.000, nilai C-Index untuk SGD masing-masing adalah 0,679 dan 0,684, sedangkan Newton-Raphson mengalami penurunan performa dengan C-Index 0,511 dan 0,551. Penelitian ini menunjukkan efektivitas SGD dalam menangkap kompleksitas data, menjadikannya pilihan yang lebih baik untuk penaksiran parameter CoxPH pada data besar.
Kata Kunci: Cox Proprtional Hazard, Log-Partial Likelihood, Newton-Raphson, Stochastic Gradient Descent, Concordance Index.

Abstract: *Parameter estimation of Cox Proportional Hazard (CoxPH) regression models often faces challenges on large datasets. In this study, the Newton-Raphson method is compared with the Stochastic Gradient Descent (SGD) method to evaluate parameter estimation. Log-partial likelihood was utilized to estimate the model parameters, and evaluated using Concordance Index (C-Index) value as the main metric. Results show that SGD is superior in all tested dataset sizes. On a dataset of 10,000 samples, SGD achieved a C-Index of 0.683, while Newton-Raphson was only 0.674. Moreover, on datasets of 50,000 and 100,000, the C-Index values for SGD were 0.679 and 0.684, respectively, while Newton-Raphson experienced a decline in performance with C-Indexes of 0.511 and 0.551. This study demonstrates the effectiveness of SGD in capturing data complexity, making it a better choice for CoxPH parameter estimation on large data.*
Keywords: *Cox Proprtional Hazard, Log-Partial Likelihood, Newton-Raphson, Stochastic Gradient Descent, Concordance Index.*

PENDAHULUAN

Dalam analisis medis, perluasan pemahaman mengenai keterkaitan antara karakteristik individu dengan risiko mengalami peristiwa medis tertentu, seperti kemajuan penyakit atau kejadian kematian, menjadi suatu aspek penting (Schober & Vetter, 2018). Dalam konteks ini, analisis data survival menjadi kritis dalam memahami permasalahan tersebut, di mana tidak hanya keberadaan peristiwa yang menjadi fokus, tetapi juga kapan peristiwa tersebut terjadi menjadi pertimbangan yang signifikan. Peristiwa yang dimaksudkan dalam hal ini dapat berupa kematian, lama sistem komputer berjalan sebelum mengalami crash, lama daya simpan produk makanan dan lainnya. Tidak hanya berupa peristiwa negatif, namun juga berupa hal positif, seperti waktu tunggu pasien dari sakit hingga sembuh (Hidayat et al., 2022)

Analisis data survival adalah serangkaian teknik statistik yang digunakan untuk meneliti data waktu sampai terjadinya suatu peristiwa (time-to-event data) dan/atau mengevaluasi hubungan antara paparan tertentu dengan terjadinya hasil tertentu setelah suatu periode pemantauan pada sekelompok individu (Abd Elhafeez et al., 2021). Namun, data survival sering kali tidak lengkap karena beberapa peristiwa mungkin belum terjadi pada akhir periode pengamatan. Hal ini disebabkan oleh

adanya data yang tersensor, di mana peristiwa yang diamati belum terjadi pada saat akhir periode pengamatan.

Salah satu metode statistik yang sering digunakan untuk analisis survival untuk mengatasi data tersensor adalah Regresi Cox Proportional Hazard. Regresi Cox Proportional Hazard (CoxPH) adalah suatu metode semiparametrik yang digunakan untuk memodelkan hubungan antara waktu bertahan (variabel respon) dengan satu atau lebih variabel prediktor. Karakteristik semiparametrik dari model ini mengindikasikan bahwa tidak ada asumsi tentang distribusi waktu survival, namun tetap mengasumsikan bahwa efek dari berbagai variabel prediktor terhadap survival tetap konstan sepanjang waktu (Schober & Vetter, 2018).

Dalam model Regresi CoxPH, parameter ditaksir dengan memaksimumkan fungsi log-konkaf yang dikenal sebagai "partial likelihood". Setelah penaksiran selesai, model tersebut dapat memberikan taksiran bahaya (hazard) khusus bagi setiap individu untuk mengalami suatu peristiwa, yang bergantung pada fitur-fitur yang dimilikinya (Tarkhan & Simon, 2020). Pendekatan umum untuk memaksimumkan partial likelihood dalam analisis regresi CoxPH adalah dengan menggunakan algoritma orde kedua yang efisien seperti Newton (Mittal et al., 2014), terutama pada dataset dengan sedikit fitur. Namun, pada data besar dengan jumlah observasi besar dan dimensi yang tinggi, penyesuaian model CoxPH dihadapi oleh beberapa masalah. Dalam penelitiannya, Simon et al. (2011) mengungkapkan bahwa penggunaan struktur sekuensial dari likelihood parsial dalam penyesuaian model Cox menyebabkan ketidakstabilan komputasi pada jumlah observasi yang besar. Hal ini menyebabkan biaya komputasi yang tinggi untuk menyesuaikan model, terutama jika data tidak cukup untuk dimuat dalam memori.

Sebuah metode diajukan dalam penelitian terkini untuk mengatasi tantangan tersebut dengan memanfaatkan stochastic gradient descent (SGD). Stochastic gradient descent merupakan metode atau algoritma optimasi yang digunakan untuk mencari nilai minimum (atau maksimum) dari sebuah fungsi objektif dengan memperbarui parameter secara iteratif dengan mengambil langkah gradien terhadap subfungsi-individu. Secara khusus, modifikasi yang diajukan memecah fungsi objektif di antara subset dari observasi, dan memungkinkan untuk menggunakan algoritma stochastic gradient descent yang melibatkan hanya subset data pada setiap iterasi (Tarkhan & Simon, 2020).

METODE PENELITIAN

Penelitian ini menggunakan pendekatan studi literatur dengan menelaah berbagai sumber kepustakaan yang relevan, termasuk buku, artikel jurnal, dan publikasi ilmiah terkait analisis survival dan metode optimasi.

HASIL DAN PEMBAHASAN

Bab ini menyajikan hasil dari penelitian yang telah dilakukan berdasarkan metode yang dijelaskan pada Bab 3. Hasil-hasil yang diperoleh akan dipaparkan melalui serangkaian evaluasi performa model Cox Proportional Hazard (CoxPH) yang dibangun menggunakan dua metode optimasi, yaitu Newton-Raphson dan *Stochastic Gradient Descent* (SGD). Selain itu, pembahasan mengenai perbandingan performa kedua metode pada dataset yang besar dengan ukuran berbeda juga akan dijelaskan. Evaluasi dilakukan dengan menggunakan Concordance Index (C-index) sebagai metrik utama untuk mengukur akurasi prediksi model.

A. Simulasi Data

Simulasi data dilakukan untuk menyediakan dataset yang digunakan dalam penaksiran dan evaluasi model Cox Proportional Hazard (CoxPH) dengan metode optimasi Newton-Raphson dan *Stochastic Gradient Descent* (SGD). Data disimulasi-kan menggunakan fungsi SimStudySACCensorConst dari package pycox dengan tingkat censoring yang ditentukan. Data yang disimulasikan mencakup tiga ukuran berbeda: 10.000, 50.000, dan 100.000 sampel, masing-masing memiliki 45 kovariat/fitur.

1. Proses Simulasi Data

Data survival (X, T, δ) disimulasikan dengan langkah-langkah sebagai berikut:

1. Pembangkitan Covariates: Sebanyak 45 kovariat $X = (X_1, X_2, \dots, X_{45})$ disimulasikan menggunakan distribusi uniform yang dimodifikasi. Dimulai dari membangkitkan bobot (weights) dari distribusi uniform (-1,1) dan setiap bobot membangkitkan covs_per_weight (kovariat). Kemudian kovariat dibangkitkan secara kondisional berdasarkan bobot dan parameter beta.
2. Pembangkitan Durasi Waktu (T): Durasi waktu hingga kejadian T disimu-lasikan menggunakan model hazard dengan fungsi hazard $h(t|X) = h_0(t)\exp(x^T\beta)$. Dibangkitkan berdasarkan kombinasi dari fungsi sinus (Simsin), fungsi hazard yang meningkat (SimAcceleratingHaz) dan fungsi hazard konstan (SimConstHaz). Dimulai dengan menghitung logit hazard untuk setiap waktu menggunakan kombinasi ketiga fungsi, mengkonversi logit hazard ke probabilitas hazard, hingga membangkitkan waktu survival berdasarkan probabilitas hazard ini.
3. Censoring: Untuk memodelkan kejadian sensor, nilai censoring disimula-sikan dengan distribusi eksponensial dengan parameter $\lambda_c = \frac{1}{30}$ yang menentukan tingkat censoring. Waktu pengamatan Y ditentukan sebagai min (T, C), dan status kejadian δ ditetapkan sebagai $\delta = I(T \leq C)$.

Tabel 1. Data Simulasi dengan ukuran 50,000 sampel

	x_1	x_2	x_3	…	x_{43}	x_{44}	x_{45}	duration	event
1	1,311	0,673	-4,867	…	0,628	0,750	-0,151	52,299	1
2	0,852	-0,148	3,166	…	-0,865	-0,373	-0,186	46,700	1
3	2,791	0,678	-2,094	…	-0,134	-1,267	0,789	42,700	1
4	0,888	-0,160	4,386	…	-0,710	0,388	-0,766	28,700	1
5	0,546	0,388	1,141	…	-0,102	-0,713	-0,106	21,299	1
…	…	…	…	…	…	…	…	…	…
49996	-1,169	-0,236	2,849	…	-0,291	0,166	-0,824	37,200	1
49997	-0,218	0,253	-3,787	…	-0,525	0,430	-0,555	49,799	1
49998	0,599	-0,455	3,325	…	0,120	-1,323	0,617	21,100	1
49999	7,189	0,211	-2,641	…	0,212	0,256	-0,740	12,200	0
50000	3,227	0,489	-1,538	…	0,781	0,311	0,533	39,900	0

2. Pra-Pemrosesan dan Analisis Data

Setelah proses simulasi data menggunakan SimStudySACCensorConst, langkah pra-pemrosesan dilakukan untuk mempersiapkan dataset bagi penaksiran dan evaluasi model. Proses ini melibatkan dua tahap utama yaitu standardisasi kovariat dan pembagian.

Tabel 2. Variabel kovariat pada data simulasi

	x_1	x_2	x_3	…	x_{43}	x_{44}	x_{45}
1	1,311	0,673	-4,867	…	0,628	0,750	-0,151
2	0,852	-0,148	3,166	…	-0,865	-0,373	-0,186
3	2,791	0,678	-2,094	…	-0,134	-1,267	0,789
4	0,888	-0,160	4,386	…	-0,710	0,388	-0,766
5	0,546	0,388	1,141	…	-0,102	-0,713	-0,106
…	…	…	…	…	…	…	…

49996	-1,169	-0,236	2,849	…	-0,291	0,166	-0,824
49997	-0,218	0,253	-3,787	…	-0,525	0,430	-0,555
49998	0,599	-0,455	3,325	…	0,120	-1,323	0,617
49999	7,189	0,211	-2,641	…	0,212	0,256	-0,740
50000	3,227	0,489	-1,538	…	0,781	0,311	0,533
Mean	**0,0175**	**-0,0004**	**0,0108**	…	**0,0035**	**0,0015**	**0,0004**
Std. Dev	**5,7152**	**0,3980**	**2,8753**	…	**0,4646**	**0,7285**	**0,4837**

Pertama, dilakukan standardisasi pada 45 variabel kovariat menggunakan Z-Score Standardization dengan rumus pada fungsi StandardScaler:

$$Z_i = \frac{X_i - \mu}{\sigma} \tag{4.1}$$

dengan μ adalah rata-rata (mean) dan σ adalah standar deviasi dari variabel kovariat X_i. Diperoleh

$$Z_1 = \frac{X_1 - 0{,}0175}{5{,}7152} \tag{4.2}$$

$$Z_2 = \frac{X_2 - (-0{,}0004)}{0{,}3980} \tag{4.3}$$

$$Z_3 = \frac{X_3 - 0{,}0108}{2{,}8753} \tag{4.4}$$

$$\vdots \qquad\qquad \vdots$$

$$Z_{43} = \frac{X_{43} - 0{,}0035}{0{,}4646} \tag{4.5}$$

$$Z_{44} = \frac{X_{44} - 0{,}0015}{0{,}7285} \tag{4.6}$$

$$Z_{45} = \frac{X_{45} - 0{,}0004}{0{,}4837} \tag{4.7}$$

Tabel 1. Variabel kovariat pada data simulasi setelah distandardisasi

	x_1	x_2	x_3	…	x_{43}	x_{44}	x_{45}
1	0,226	1,693	-1,696	…	1,345	1,028	-0,314
2	0,146	-0,371	1,097	…	-1,871	-0,514	-0,386
3	0,485	1,705	-0,732	…	-0,297	-1,742	1,631
4	0,152	-0,401	1,521	…	-1,536	0,530	-1,585
5	0,092	0,977	0,393	…	-0,227	-0,981	-0,220
…	…	…	…	…	…	…	…
49996	-0,207	-0,592	0,987	…	-0,634	0,225	-1,701
49997	-0,041	0,636	-1,320	…	-1,138	0,589	-1,149
49998	0,101	-1,142	1,152	…	0,251	-1,819	1,275
49999	1,254	0,531	-0,922	…	0,449	0,349	-1,532
50000	0,561	1,230	-0,538	…	1,673	0,425	1,102

Setelah distandardisasi, dataset dibagi menjadi tiga bagian: data latih, validasi dan pengujian. Sebanyak 80% dari total data digunakan sebagai data latih, sementara 20% sisanya dibagi sama rata antara data validasi dan pengujian. Pembagian ini dilakukan secara acak untuk memastikan representasi yang seimbang dari populasi data pada setiap dataset. Perlu dicatat bahwa meskipun data validasi disediakan, penggunaannya hanya spesifik untuk metode SGD yang menggunakan pycox dalam pelatihan model jaringan saraf, sementara metode Newton yang menggunakan lifelines tidak memanfaatkan data validasi dalam prosesnya.

Tabel 4. Jumlah data pada setiap ukuran dataset

Ukuran Dataset	Data Latih	Data Validasi	Data Uji
10,000	8.000	1.000	1.000
50.000	40.000	5.000	5.000
100.000	80.000	10.000	10.000

Proses pra-pemrosesan ini memastikan bahwa data siap digunakan untuk penaksiran dan evaluasi model Cox Proportional Hazard (CoxPH) dengan kedua optimasi yang akan dibandingkan, yaitu Newton-Raphson dan Stochastic Gradient Descent (SGD), dengan mempertimbangkan kebutuhan spesifik masing-masing metode.

B. Penaksiran Model Regresi Cox Proportional Hazard (CoxPH)

Pada bagian ini, akan dipaparkan proses penaksiran parameter model regresi CoxPH pada persamaan 2.3 menggunakan dua metode optimasi: Newton Raphson dan Stochastic Gradient Descent. Proses ini bertujuan untuk memaksimumkan fungsi log-partial likelihood atau meminimumkan fungsi negatif log-partial likelihood pada persamaan 2.6. Karena data simulasi mengandung informasi tentang adanya data yang tersensor, maka fungsi log-partial likelihood yang digunakan sebagai fungsi loss perlu disesuaikan dengan mempertimbangkan pensensoran sebagai berikut:

$$loss = -\sum_{i=1}^{n} \delta_i \left(f_\beta\left(x^{(i)}\right) - \log\left(\sum_{j \in \mathcal{R}_i} \exp\left(f_\beta\left(x^{(j)}\right)\right)\right)\right) \tag{4.8}$$

dimana, δ_i adalah indicator kejadian yang bernilai 1 jika peristiwa terjadi dan 0 jika tersensor.

1. Penaksiran dengan Metode Newton Raphson

Berikut akan dijelaskan proses penaksiran parameter model regresi CoxPH menggunakan metode optimasi Newton-Raphson. Perhitungan manual pembaruan nilai parameter dilakukan menggunakan data simulasi berukuran 50.000 sampel, yang telah melalui pra-pemrosesan dan analisis data. Penaksiran parameter ini dimple-mentasikan menggunakan kelas CoxPHFitter dari paket lifelines. *Source Code* untuk melakukan penaksiran parameter terdapat pada lampiran.

Tabel 2. Data Latih (x_{train}) pada ukuran dataset 50.000

	x_1	x_2	x_3	...	x_{43}	x_{44}	x_{45}	duration	event
1	0,455	0,752	0,429	...	0,891	-0,386	0,609	25,700	1
2	1,279	1,996	-0,405	...	0,668	0,331	-0,886	26,000	1
3	-0,766	-0,474	0,671	...	-0,178	0,468	-0,380	4,800	1
4	0,454	0,156	-0,654	...	-0,372	-1,076	0,569	23,100	1
5	-0,845	-0,519	-0,707	...	-1,441	-0,229	-0,455	43,000	1
...	...	...	...	...	...	...	...	...	...
39996	0,310	0,662	-0,682	...	-1,791	-0,774	-0,506	6,800	1
39997	1,495	1,045	-0,440	...	-1,553	-1,032	0,664	60,299	1
39998	1,076	2,068	-1,220	...	1,661	1,617	-0,936	88,800	0
39999	1,093	1,886	-0,319	...	-0,070	0,375	-1,348	47,299	1
40000	-2,267	-2,107	0,456	...	0,357	-0,038	0,688	58,299	0

Langkah 1: Inisialisasi Parameter

Pertama, diinisalisasikan vektor parameter β secara acak. Dikarenakan data simulai memiliki 45 kovariat, maka:

$$\hat{\beta}(0) = \left[\hat{\beta}_1(0), \hat{\beta}_2(0), \dots, \hat{\beta}_{45}(0)\right]^T \tag{4.9}$$

Dalam implementasinya, digunakan inisialisasi:

$$\hat{\beta}(0) = [0{,}000 \quad 0{,}000 \quad \dots \quad 0{,}000]^T$$

Langkah 2: Proses Iteratif

Selanjutnya, dilakukan proses iteratif sesuai dengan persamaan 2.7 hingga mencapai konvergensi. Untuk setiap iterasi m:

a. Menghitung gradien dari fungsi log-partial likelihood

Pada persamaan 8. diberikan fungsi log-partial likelihood yang digunakan sebagai fungsi loss sebagai berikut:

$$loss = -\sum_{i=1}^{n} \delta_i \left(f_\beta\left(x^{(i)}\right) - \log\left(\sum_{j \in \mathcal{R}_i} \exp\left(f_\beta\left(x^{(j)}\right)\right)\right)\right)$$

- Hitung turunan dari bagian pertama:
  Turunan dari $f_\beta\left(x^{(i)}\right) = x^{(i)T}\beta$ terhadap β adalah $x^{(i)}$.
- Hitung turunan dari bagian log-sum-eksponensial:
  Turunan dari $\log\left(\sum_{j \in \mathcal{R}_i} \exp\left(f_\beta\left(x^{(j)}\right)\right)\right)$ terhadap β menggunakan aturan rantai:

$$\frac{\partial}{\partial \beta} \log\left(\sum_{j \in \mathcal{R}_i} \exp\left(f_\beta\left(x^{(j)}\right)\right)\right) = \frac{\sum_{j \in \mathcal{R}_i} \mathrm{x}^{(j)} \exp\left(f_\beta\left(x^{(j)}\right)\right)}{\sum_{j \in \mathcal{R}_i} \exp\left(f_\beta\left(x^{(j)}\right)\right)} \quad (4.10)$$

- Maka turunan dari fungsi log-partial likelihood adalah:

$$\frac{\partial loss}{\partial \beta} = -\sum_{i=1}^{n} \delta_i \left(x^{(i)} - \frac{\sum_{j \in \mathcal{R}_i} \mathrm{x}^{(j)} \exp\left(f_\beta\left(x^{(j)}\right)\right)}{\sum_{j \in \mathcal{R}_i} \exp\left(f_\beta\left(x^{(j)}\right)\right)}\right) \quad (4.11)$$

Berikut perhitungan gradien dari fungsi log-partial likelihood dari data latih ($\mathrm{x}_{\mathrm{train}}$) pada Tabel 5:

Untuk $i = 1$; $x^{(1)} = [0{,}455 \quad 0{,}752 \quad \ldots \quad 0{,}609]$:

- Identifikasi δ_i:

$$\delta_i = 1$$

- Hitung $f_\beta\left(x^{(1)}\right) = x^{(1)T}\hat{\beta}(0)$

$$= 0{,}455 \cdot 0 + 0{,}752 \cdot 0 + \cdots + (-0{,}386) \cdot 0 + 0{,}609 \cdot 0$$
$$= 0$$

- Hitung himpunan risiko $\mathcal{R}_i$ berdasarkan durasi (*duration*):

$$\mathcal{R}_i = \{j : duration_j \geq duration_i\} \quad (4.12)$$

di mana:
$duration_{i=1} = 25{,}700$
Sehingga untuk $\sum_{j \in \mathcal{R}_i} \exp\left(f_\beta\left(x^{(j)}\right)\right)$ diperoleh:

$$\sum_{j \in \mathcal{R}_i} \exp\left(f_\beta\left(x^{(j)}\right)\right) = \sum_{j \in \mathcal{R}_i} \exp(0) = \sum_{j \in \mathcal{R}_i} 1$$

Ini menghasilkan jumlah n_i, jumlah subjek dalam himpunan risiko $\mathcal{R}_i$ pada $i = 1$.

- Hitung $\sum_{j \in \mathcal{R}_i} \mathrm{x}^{(j)} \exp\left(f_\beta\left(x^{(j)}\right)\right)$, sama seperti sebelumnya:

$$\sum_{j \in \mathcal{R}_i} \mathrm{x}^{(j)} \exp\left(f_\beta\left(x^{(j)}\right)\right) = \sum_{j \in \mathcal{R}_i} \mathrm{x}^{(j)}$$

adalah jumlah dari $x^{(j)}$ untuk semua j dalam $\mathcal{R}_i$.

- Substitusi ke dalam formula gradien:

$$\frac{\partial loss}{\partial \beta} = -\left(1 \times \left(0{,}455 - \frac{\sum_{j \in \mathcal{R}_i} \mathrm{x}^{(j)}}{n_1}\right)\right) \quad (4.13)$$

Diperoleh nilai gradien dari fungsi log-partial likelihood dalam bentuk vektor gradien adalah:

$$\begin{bmatrix} 6{,}597 \\ -9{,}016 \\ \vdots \\ -1{,}316 \end{bmatrix}$$

b. Menghitung matriks Hessian dari fungsi log-partial likelihood

Untuk mendapatkan matriks Hessian, ambil turunan dari persamaan 4.11 terhadap β:

$$\frac{\partial^2 loss}{\partial \beta^2} = -\sum_{i=1}^{n} \delta_i \frac{\partial}{\partial \beta}\left(x^{(i)} - \frac{\sum_{j\in\mathcal{R}_i} \mathrm{x}^{(j)} \exp\left(f_\beta\left(x^{(j)}\right)\right)}{\sum_{j\in\mathcal{R}_i} \exp\left(f_\beta(x^{(j)})\right)}\right) \tag{4.14}$$

- Bagian pertama: $x^{(i)}$

Turunan terhadap β adalah nol karena $x^{(i)}$ adalah kovariat.

- Bagian kedua: Turunan Logaritma

Gunakan aturan hasil bagi untuk menghitung turunan dari:

$$\frac{\partial}{\partial \beta}\left(\frac{\sum_{j\in\mathcal{R}_i} \mathrm{x}^{(j)} \exp\left(f_\beta\left(x^{(j)}\right)\right)}{\sum_{j\in\mathcal{R}_i} \exp\left(f_\beta(x^{(j)})\right)}\right)$$

Aturan hasil bagi menyatakan bahwa:

$$\frac{\partial}{\partial \beta}\left(\frac{A}{B}\right) = \frac{B\frac{\partial A}{\partial \beta} - A\frac{\partial B}{\partial \beta}}{B^2} \tag{4.15}$$

di mana:

$A = \sum_{j\in\mathcal{R}_i} \mathrm{x}^{(j)} \exp\left(f_\beta\left(x^{(j)}\right)\right)$

$B = \sum_{j\in\mathcal{R}_i} \exp\left(f_\beta\left(x^{(j)}\right)\right)$

Dengan menggunakan definisi dari A dan B:

$$A' = \sum_{j\in\mathcal{R}_i} \mathrm{x}^{(j)} \exp\left(f_\beta\left(x^{(j)}\right)\right) \frac{\partial f_\beta(x^{(j)})}{\partial \beta}$$

$$B' = \sum_{j\in\mathcal{R}_i} \exp\left(f_\beta\left(x^{(j)}\right)\right) \frac{\partial f_\beta(x^{(j)})}{\partial \beta}$$

- Dengan menyusun semua komponen, dapat diekspresikan matriks Hessian sebagai:

$$H_{\beta_i\beta_j} = \sum_{i=1}^{n} \delta_i \left(\frac{\partial^2}{\partial \beta^2}(loss)\right)$$

$$H_{\beta_i\beta_j} = \sum_{i=1}^{n} \delta_i \left(\frac{\partial^2}{\partial \beta^2}\left(-log\left(\sum_{j\in\mathcal{R}_i} \exp\left(f_\beta\left(x^{(j)}\right)\right)\right)\right)\right) \tag{4.16}$$

Maka elemen-elemen dari matriks Hessian H didefinisikan sebagai

$$H_{\beta_i\beta_j} = \frac{\partial^2 loss}{\partial \beta_i \partial \beta_j} \tag{4.17}$$

Sehingga, matriks Hessian yang berisi turunan kedua dari fungsi loss terhadap setiap pasangan parameter β_i dan β_j dapat dituliskan dalam bentuk:

$$H = \begin{bmatrix} \frac{\partial^2 loss}{\partial \beta_1^2} & \frac{\partial^2 loss}{\partial \beta_1 \partial \beta_2} & \cdots \\ \frac{\partial^2 loss}{\partial \beta_2 \partial \beta_1} & \frac{\partial^2 loss}{\partial \beta_2^2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \quad (4.18)$$

Diperoleh nilai Hessian dari fungsi log-partial likelihood dalam bentuk matriks adalah:

$$\begin{bmatrix} -2{,}507 & -1{,}237 & \cdots & -1{,}223 \\ -1{,}237 & -2{,}485 & \cdots & 1{,}069 \\ \vdots & \vdots & \ddots & \vdots \\ -1{,}223 & 1{,}069 & \cdots & -2{,}499 \end{bmatrix}$$

c. Memperbarui nilai parameter

Setelah menghitung gradien dan matriks Hessian, nilai parameter $\hat{\beta}$ diperbarui menggunakan persamaan 2.7 yaitu:

$$\hat{\beta}(m+1) = \hat{\beta}(m) - \left[H\left(\hat{\beta}(m)\right)\right]^{-1} \nabla(loss)$$

Dengan menggunakan informasi matriks Hessian(H_{ij}) dan vektor gradien (∇(loss)) yang berisi komponen-komponen hessian dan gradien untuk setiap parameter β, maka pembaruan nilai parameter $\hat{\beta}$ menjadi:

$$\hat{\beta}(1) = \begin{bmatrix} 0{,}000 \\ 0{,}000 \\ \vdots \\ 0{,}000 \end{bmatrix} - \begin{bmatrix} -2{,}507 & -1{,}237 & \cdots & -1{,}223 \\ -1{,}237 & -2{,}485 & \cdots & 1{,}069 \\ \vdots & \vdots & \ddots & \vdots \\ -1{,}223 & 1{,}069 & \cdots & -2{,}499 \end{bmatrix}^{-1} \begin{bmatrix} 6{,}597 \\ -9{,}016 \\ \vdots \\ -1{,}316 \end{bmatrix}$$

Langkah 3: Konvergensi

Langkah-langkah di atas dilakukan secara iteratif untuk seluruh data hingga mencapai konvergensi. Proses iterasi yang dilakukan oleh CoxPHFitter dari paket lifelines ditunjukkan dalam tabel berikut:

Tabel 3 Proses Iterasi Newton Raphson

Iterasi	Norm Delta	Step Size	Log Likelihood	Newton Decrement	Waktu
1	$1{,}97e+00$	0,950	-246839,41131	$4{,}62e+03$	0,3
2	$1{,}10e-01$	0,950	-242242,58446	$1{,}57e+01$	0,5
3	$5{,}85e-03$	0,950	-242226,92774	$4{,}36e-02$	0,7
4	$1{,}05e-06$	1,000	-242226,88412	$1{,}39e-09$	1,0

Metode Newton-Raphson memperbarui nilai parameter hingga perubahan antar iterasi (Norm Delta) menjadi sangat kecil dan nilai Newton Decrement mendekati nol. Peningkatan nilai log-likelihood pada setiap iterasi menunjukkan bahwa model semakin sesuai dengan data yang diberikan, menunjukkan konvergensi parameter $\hat{\beta}$ yang semakin optimal.

Sehingga parameter $\hat{\beta}$ pada persamaan 4.9 yang telah ditaksir adalah:

$$\hat{\beta} = [0{,}019 \quad -0{,}006 \quad \ldots \quad -0{,}323]^T$$

Nilai-nilai parameter ini menggambarkan kontribusi masing-masing kovariat terhadap risiko kejadian dalam model CoxPH. Selain itu, hasil taksiran nilai *baseline hazard* (h_0) dapat dilihat pada lampiran.

2. Penaksiran dengan Metode Stochastic Gradient Descent

Berikut akan dijelaskan proses penaksiran parameter model regresi CoxPH menggunakan metode optimasi Stochastic Gradient Descent (SGD). Metode ini

memanfaatkan subset data (*batch*) yang dipilih secara acak pada setiap iterasi. Perhitungan manual pembaruan nilai parameter dilakukan menggunakan data simulasi pada Tabel 4.5. Penaksiran parameter ini diimplementasikan menggunakan kelas CoxPH dari paket pycox dengan memanfaatkan kekuatan jaringan saraf dalam Pytorch untuk melakukan penaksiran parameter dalam model CoxPH. *Source Code* untuk melakukan penaksiran parameter terdapat pada lampiran.

Langkah 1: Inisialisasi Parameter

Dalam implementasinya, parameter $\hat{\beta}$ diinisialisasikan secara acak dengan nilai awal $\hat{\beta}(0)$. Inisialisasi ini dilakukan menggunakan distribusi seragam:

$$\mathcal{U}(-\frac{1}{\sqrt{in\_features}}, \frac{1}{\sqrt{in\_features}}) \quad (4.19)$$

di mana in_features adalah jumlah fitur input sebesar 45 yang merupakan kovariat dari data simulasi, maka nilai $k = \frac{1}{\sqrt{45}} \approx 0{,}149$. Sehingga parameter $\hat{\beta}(0)$ diinisialisasi secara acak dalam interval $[-0{,}149\ ;\ 0{,}149]$:

$$\hat{\beta}(0) = [0{,}100 \quad -0{,}100 \quad 0{,}050 \quad \dots \quad -0{,}100]^T$$

Langkah 2: Tetapkan Laju Pembelajaran (η)

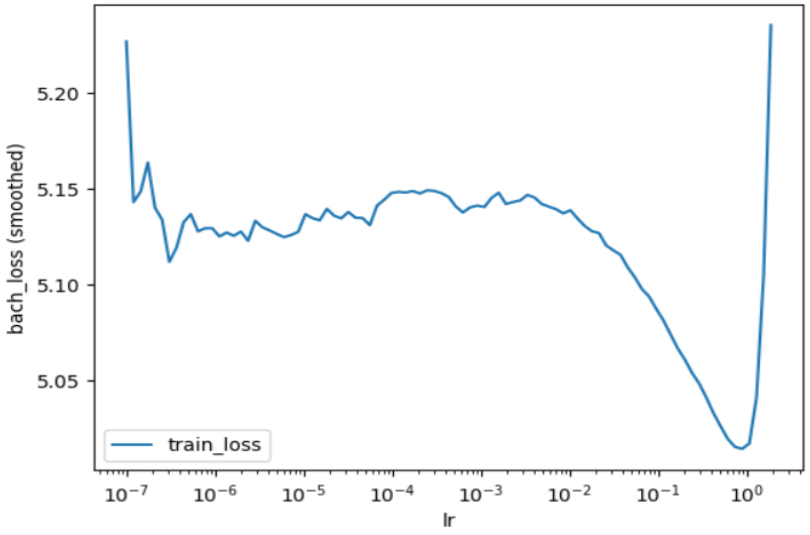
Laju pembelajaran ditetapkan menggunakan metode lrfinder dari paket pytorch untuk mendapatkan laju pembelajaran yang optimal. (Smith, 2015) menje- laskan bahwa pencarian laju pembelajaran ini melakukan percobaan kecil di mana laju pembelajaran (*learning rate*) secara bertahap ditingkatkan secara eksponensial setelah setiap *batch* yang diproses, dan loss yang sesuai dicatat.

Dalam proses ini, *learning rate* diinisialisasi pada nilai kecil yaitu 10^{-7}, hingga mencapai 10^{0} dalam skala eksponensial.

$$lr(t) = start\_lr \times \left(\frac{end\_lr}{start\_lr}\right)^{t/T} \quad (4.20)$$

di mana T adalah jumlah total iterasi (*batch*). Hasilnya adalah grafik lr vs. loss yang dapat digunakan sebagai panduan untuk memilih laju pembelajaran awal yang optimal.

Grafik hasil lrfinder yang menunjukkan laju pembelajaran terbaik disajikan pada Gambar 4.1.



Gambar 1 Grafik penentuan laju pembelajaran optimal

Grafik menunjukkan bahwa loss menurun secara signifikan saat laju pembelajaran berada di sekitar 10^{-1}. Namun, dalam implementasi dipilih laju pembelajaran $\eta = 0{,}01$,

untuk mempertahankan keseimbangan antara kecepatan pembelajaran dan stabilitas proses pelatihan, guna menghindari *overfitting* pada model.

Langkah 3: Tetapkan Ukuran *Batch* dan Ukuran Himpunan Risiko

- Ukuran *batch B* yang digunakan adalah 256
- Himpunan risiko $|\widetilde{R}_\iota|$ adalah subset dari data *batch B*, mencakup semua individu dengan durasi kejadian yang lebih panjang atau sama dengan individu i

$$\left|\widetilde{R}_\iota\right| = \{j \in B | duration_j \geq duration_i\} \qquad (4.21)$$

Langkah 4: Proses Iteratif

Proses iteratif dilakukan hingga model mencapai konvergensi. Untuk setiap iterasi m:

a. Subsampling data ke dalam *batch* B

Pilih ukuran *batch* $B = 256$. Ambil subset dari data latih ($\mathrm{x}_{\mathrm{train}}$) pada Tabel 4.5 secara acak dengan ukuran *batch B*.

$$Batch = \{x^{(1)}, x^{(2)}, \dots, x^{(256)}\} \qquad (4.22)$$

Tabel 4 Sampel *Batch* Acak dari Data Latih x_{train}

	x_1	x_2	x_3	…	x_{43}	x_{44}	x_{45}	duration	event
1	-0,518	-0,793	0,741		-0,460	0,499	-1,605	24,799	1
2	0,251	-0,119	0,877		-0,039	0,085	-1,385	38,800	0
3	1,210	0,133	-0,676		-0,508	-2,253	1,684	35,500	1
4	0,827	0,029	0,455		-0,009	-0,069	0,619	41,900	1
5	-0,970	-0,271	-0,673		0,806	0,773	0,014	87,599	0
…	…	…	…		…	…	…	…	…
252	-1,532	-1,130	0,396		-1,734	-1,735	1,408	9,200	1
253	0,491	-0,559	0,666		0,086	-0,256	1,023	100,000	0
254	-0,347	-0,602	-1,745		1,387	1,108	-0,086	27,900	1
255	-0,350	1,479	-0,575		0,779	0,685	-0,041	71,300	0
256	1,401	1,136	-0,058		-0,140	-0,916	-0,154	12,400	0

b. Untuk setiap individu i dalam *batch* B

Untuk setiap individu i dalam *batch B* akan melibatkan perhitungan gradien dan pembaruan parameter model.

Untuk $i = 1$; $x^{(1)} = [-0{,}518 \quad -0{,}793 \quad \dots \quad -1{,}605]$:

- Bentuk Himpunan Risiko $|\widetilde{R}_\iota|$ yang mencakup individu j dalam *batch B* dengan $duration_j \geq duration_i = 24{,}799$.

Berdasarkan sampel *batch* pada Tabel 4.7, individu-individu yang memiliki $duration_j \geq duration_i = 24{,}799$ adalah:

$$\{i^{(1)}(24{,}799), i^{(2)}(38{,}800), i^{(3)}(35{,}500), \dots, i^{(254)}(27{,}900), i^{(256)}(71{,}300)$$

Jadi, himpunan $\widetilde{R}_\iota$ untuk individu pertama akan mencakup individu-individu berikut:

$$\widetilde{R}_\iota = \{1{,}2{,}3, \dots, 254{,}255\}$$

- Hitung gradien fungsi log-partial likelihood

Untuk individu i gradien dari fungsi log-partial likelihood pada persamaan 4.11 akan menjadi:

$$\nabla loss_i = x^{(i)} - \frac{\sum_{j \in \mathcal{R}_i} \mathrm{x}^{(j)} \exp\left(f_\beta\left(x^{(j)}\right)\right)}{\sum_{j \in \mathcal{R}_i} \exp\left(f_\beta(x^{(j)})\right)} \qquad (4.23)$$

Di sini himpunan risiko $\widetilde{R}_\iota = \{1{,}2{,}3, \dots, 254{,}255\}$.

c. Pembaruan parameter $\hat{\beta}$

Setelah gradien loss dihitung menggunakan persamaan 4.11, parameter $\hat{\beta}$ diperbarui

menggunakan metode SGD menggunakan persamaan 2.10:

$$\hat{\beta}(m+1) = \hat{\beta}(m) - \eta \cdot \nabla(loss)$$

Dengan menggunakan informasi laju pembelajaran η dan gradien dari fungsi log-partial likelihood, maka pembaruan nilai parameter $\hat{\beta}$ menjadi:

$$\hat{\beta}(1) = \begin{bmatrix} 0{,}100 \\ -0{,}100 \\ \vdots \\ -0{,}100 \end{bmatrix} - 0{,}01 \cdot \nabla loss_i$$

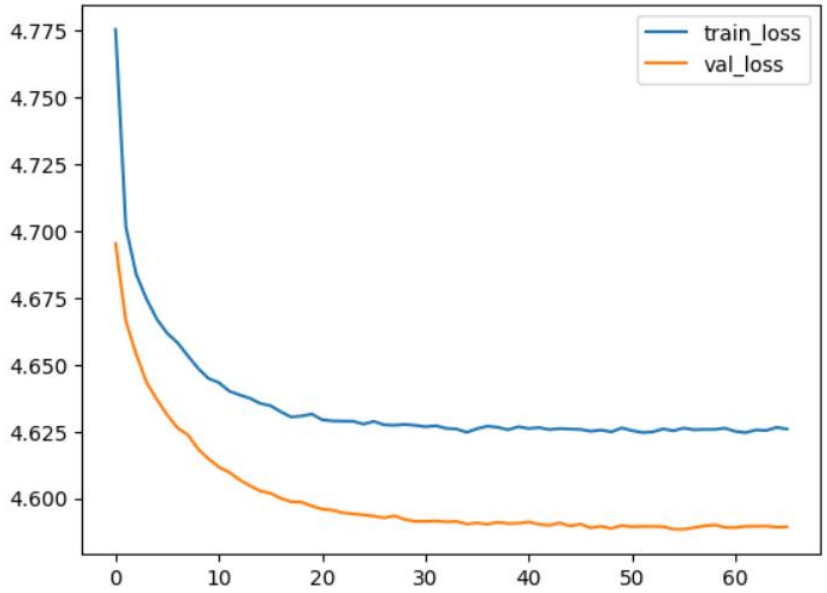
Langkah 5: Konvergensi

Langkah-langkah di atas dilakukan hingga parameter $\hat{\beta}$ mencapai konvergensi, yaitu ketika perubahan nilai loss pada data validasi menunjukkan penurunan yang sta-bil. Proses ini menggambarkan kemampuan model dalam memprediksi hasil pada data yang belum pernah dilihat sebelumnya (data validasi). Proses iterasi dilakukan oleh menggunakan paket pycox dengan pendekatan jaringan saraf oleh dalam Pytorch.

Tabel 5 Proses Iterasi Stochastic Gradient Descent (SGD)

Iterasi	Loss Latih	Loss Validasi	Waktu
1	4,775	4,695	0,0
2	4,701	4,666	0,0
3	4,683	4,653	0,0
...	...	...	...
64	4,625	4,589	16,0
65	4,626	4,589	16,0
66	4,625	4,589	17,0

Dari hasil di atas, terlihat bahwa nilai loss pada data validasi menurun secara konsisten, yang menunjukkan bahwa model semakin baik dalam memprediksi hasil pada data yang baru. Grafik di bawah ini menunjukkan kurva penurunan loss selama pelatihan menggunakan metode SGD:



Gambar 2 Kurva penurunan loss menggunakan metode SGD

Setelah proses pelatihan selesai, parameter $\hat{\beta}$ yang telah ditaksir adalah:

$$\hat{\beta} = [0{,}020 \quad -0{,}006 \quad \ldots \quad -0{,}3074]^T$$

Nilai parameter menggambarkan kontribusi masing-masing kovariat terhadap risiko kejadian dalam model CoxPH. Semakin besar nilai positif, maka semakin tinggi risiko kejadian terkait dengan kovariat tersebut, begitu pula sebaliknya. Selain itu hasil taksiran

nilai *baseline hazard* (h_0) yang memberi informasi mengenai risiko dasar tersedia dalam lampiran.

C. Evaluasi Model

Setelah model Cox Proportional Hazard (CoxPH) dibangun dan parameter β ditaksir, dilakukan evaluasi model untuk menilai performa prediksi risiko kejadian dari masing-masing metode pada data uji. Evaluasi ini menggunakan Concordance Index (C-Index) sebagai metrik utama dan merupakan salah satu metrik paling umum dalam analisis survival. C-Index menilai seberapa baik model yang telah dilatih mampu memprediksi urutan kejadian pada data yang belum pernah dilihat sebelumnya (data uji).

Tabel 6 Data Uji (x_{test}) pada ukuran dataset 50.000

	x_1	x_2	x_3	...	x_{43}	x_{44}	x_{45}	duration	event
1	1,311	0,673	-4,867		0,628	0,750	-0,151	52,299	1
2	0,266	-0,089	-5,137		0,453	0,396	-0,539	79,000	0
3	4,505	0,484	2,152		0,077	-0,278	0,409	2,700	1
4	6,437	-0,307	3,294		-0,560	-0,809	0,297	6,200	1
5	5,814	0,203	-1,968		-0,290	-0,925	0,595	0,700	1
...	...	...	...		...	...	...	...	...
4996	1,510	-0,612	5,323		-0,198	0,814	-0,455	100,000	0
4997	2,351	0,175	-0,828		-0,457	0,518	-0,688	0,100	1
4998	-3,622	0,123	2,760		0,437	1,312	-0,809	17,200	1
4999	-1,847	0,166	0,648		-0,238	-0,673	-0,005	11,900	1
5000	3,192	0,114	3,622		0,326	0,959	-0,392	12,800	1

C-Index dinyatakan dengan rumus yang telah dijelaskan pada persamaan 2.11:

$$C_{index} = \frac{\sum_{i,j} 1\{T_j < T_i\} \cdot 1\{r_j > r_i\} \cdot \delta_j}{\sum_{i,j} 1\{T_j < T_i\} \cdot \delta_j}$$

Nilai C-Index berkisar antara 0 hingga 1:

- $C_{index} = 1$: menunjukkan model memiliki prediksi yang sempurna
- $C_{index} = 0{,}5$: menunjukkan model memiliki prediksi yang acak
- $C_{index} < 0{,}5$: menunjukkan model memiliki prediksi yang buruk

Parameter $\hat{\beta}$ yang telah ditaksir digunakan untuk menghitung skor risiko r_j bagi setiap individu dalam data uji. Skor risiko r_j dihitung sebagai kombinasi linear dari kovariat x_j dan parameter yang telah ditaksir $\hat{\beta}$, yaitu:

$$r_j = x_j \cdot \hat{\beta} \qquad (4.24)$$

di sini x_j adalah vektor kovariat untuk individu ke- j.

Dengan menggunakan parameter yang telah ditaksir dengan metode Newton-Raphson $\hat{\beta} = [0{,}019 \quad -0{,}006 \quad \dots \quad -0{,}323]$, dan vektor kovariat $x_j = [1{,}311 \quad 0{,}673 \quad -3{,}867 \quad \dots \quad -0{,}151]$ maka skor risiko untuk individu ke -1:

$r_1 = x_1 \cdot \hat{\beta}_1 + x_2 \cdot \hat{\beta}_2 + x_3 \cdot \hat{\beta}_3 + \cdots + x_{45} \cdot \hat{\beta}_{45}$

$r_1 = (1{,}311 \cdot 0{,}019) + (0{,}673 \cdot -0{,}006) + \cdots + (-0{,}151 \cdot -0{,}323)$

Setelah menghitung skor risiko untuk semua individu, bandingkan pasangan individu yang hanya memiliki $\delta_j = 1$. Periksa apakah skor risiko sesuai dengan urutan waktu kejadian ($duration$). Jika $duration_j < duration_i$ dan $r_j > r_i$, maka pasangan tersebut *concordant*. Sehingga persamaan 2.11 akan menjadi:

$$C_{index} = \frac{Jumlah\ concordant}{Total\ Pasangan\ yang\ Dibandingkan} \qquad (4.25)$$

Dengan demikian diperoleh nilai C-Index model CoxPH pada data uji dengan 50.000 sampel adalah

- Metode Newton-Raphson :$C_{index} = 0{,}511$
- Metode SGD :$C_{index} = 0{,}679$

D. Analisis Hasil

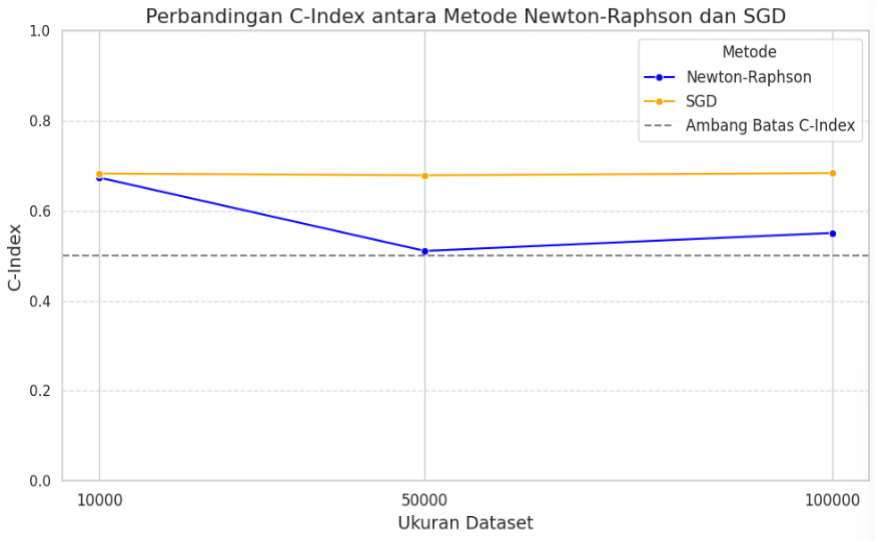
Analisis hasil penelitian ini membahas performa kedua metode optimasi, yaitu Newton-Raphson dan Stochastic Gradient Descent (SGD), dalam model CoPH. Hasil evaluasi disajikan dalam bentuk nilai Concordance Index (C-Index), yang berfungsi untuk mengukur kemampuan model dalam memprediksi urutan kejadian berdasarkan data uji (x_{test}).

Pada skenario dengan ukuran dataset yang berbeda, perbandingan hasil *C-Index* yang diperoleh dari kedua metode ditampilkan dalam Tabel 4.10 berikut:

Tabel 7 Hasil Evaluasi C-Index pada Model CoxPH menggunakan Metode Newton-Raphson dan SGD

Metode	Ukuran Dataset	C-Index
Newton-Raphson	10.000	0,674
	50.000	0,511
	100.000	0,551
SGD	10.000	0,683
	50.000	0,679
	100.000	0,684

Dari tabel di atas, terlihat bahwa metode Stochastic Gradient Descent (SGD) secara konsisten menghasilkan performa yang lebih baik dibandingkan dengan metode Newton-Raphson dalam memprediksi kejadian pada semua ukuran dataset. Pada dataset berukuran 10.000, SGD menunjukkan keunggulan dalam menangkap pola data dengan nilai C-Index yang lebih tinggi dibandingkan dengan Newton-Raphson.



Gambar 3 Perbandingan C-Index Metode Newton Raphson dan SGD

Sementara itu, pada dataset yang lebih besar, yaitu 50.000 dan 100.000, metode Newton-Raphson mengalami penurunan performa yang signifikan, terlihat dari nilai C-Index yang menurun. Hal ini menunjukkan bahwa metode Newton-Raphson kesulitan dalam menangkap kompleksitas pola yang ada dalam data saat ukuran dataset meningkat. Sebaliknya, C-Index untuk SGD tetap stabil dan bahkan meningkat pada ukuran dataset yang lebih besar, menandakan efektivitas dan ketahanannya dalam penaksiran parameter model CoxPH pada data besar.

Keunggulan Stochastic Gradient Descent (SGD) dalam menangani dataset besar sangat penting, terutama karena metode ini menggunakan subset data untuk

memperbarui parameter model secara bertahap. Dengan pembaruan parameter yang dilakukan secara iteratif melalui subset data, SGD menjadi lebih efisien dalam menangani kompleksitas data survival dibandingkan metode Newton-Raphson. Selain itu, pengaturan hyperparameter seperti laju pembelajaran dan ukuran *batch* dapat disesuaikan dengan ukuran dataset untuk meningkatkan kinerja model, sebagai contoh pada dataset dengan 10.000 sampel, laju pembelajaran sebesar 0,01 dan ukuran *batch* 128 dapat diterapkan. Di sisi lain, metode Newton Raphson cenderung mengalami penurunan performa ketika ukuran dataset semakin besar, yang menunjukkan batasan dalam menangani data besar.

KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan bahwa metode ini menunjukkan performa yang baik, terutama dalam menangani dataset besar. Perbandingan dengan metode Newton-Raphson mengungkapkan bahwa SGD menghasilkan nilai Concordance Index (C-Index) yang lebih tinggi di berbagai ukuran dataset. Pada dataset 10.000 sampel, SGD mencapai C-Index 0,683, sedangkan Newton-Raphson hanya 0,674; bahkan hingga pada dataset 100.000 sampel, SGD tetap unggul dengan nilai 0,684, sementara Newton-Raphson hanya 0,551. Keunggulan ini terlihat dari kemampuan SGD untuk melakukan pembaruan parameter model secara efisien menggunakan subset data. Sementara itu Newton-Raphson mengalami penurunan yang signifikan pada ukuran dataset yang lebih besar, menunjukkan keterbatasannya dalam menangani kompleksitas data. Dengan pengaturan hyperparameter yang tepat, SGD mampu mempertahankan kinerja yang baik, menjadikannya pilihan yang lebih baik untuk analisis survival dengan data besar.

Saran

Berdasarkan hasil yang diperoleh, beberapa saran yang dapat dipertimbangkan untuk penelitian selanjutnya adalah sebagai berikut:

1. Penelitian ini menunjukkan bahwa pengaturan hyperparameter, seperti laju pembelajaran dan ukuran batch, sangat mempengaruhi performa model, sehingga penelitian ke depan bisa mencoba metode otomatisasi untuk mendapatkan hasil yang lebih optimal dalam berbagai skenario dataset.
2. Untuk meningkatkan stabilitas model dan mengurangi risiko overfitting, disarankan untuk mengeksplorasi teknik regularisasi seperti L1/L2 Regularization atau dropout, terutama pada dataset dengan dimensi yang sangat besar.
3. Selain itu, untuk memperkaya hasil penelitian, melakukan studi perbandingan antara SGD dengan metode optimasi lainnya, seperti Adam, AdaGrad, atau RMSProp, akan memberikan perspektif yang lebih luas tentang efektivitas berbagai pendekatan dalam penaksiran parameter model CoxPH.

DAFTAR PUSTAKA

Abd Elhafeez, S., D'Arrigo, G., Leonardis, D., Fusaro, M., Tripepi, G., & Roumeliotis, S. (2021). Methods to Analyze Time-to-Event Data: The Cox Regression Analysis. Oxidative Medicine and Cellular Longevity, 2021. https://doi.org/10.1155/2021/1302811

Fox, J., & Weisberg, S. (n.d.). Cox Proportional-Hazards Regression for Survival Data in R. http://tinyurl.com/carbook

Harrell, F. E., Lee, K. L., & Mark, D. B. (1996). TUTORIAL IN BIOSTATISTICS MULTIVARIABLE PROGNOSTIC MODELS: ISSUES IN DEVELOPING MODELS, EVALUATING ASSUMPTIONS AND ADEQUACY, AND MEASURING AND REDUCING ERRORS. In STATISTICS IN MEDICINE (Vol. 15).

Hidayat, R., Sam, M., Wardi, R. Y., & Iskandar, M. I. (2022). Pemodelan Survival Pasien Covid-19 dengan Hazard Non-Proporsional. Euler : Jurnal Ilmiah Matematika, Sains Dan Teknologi, 10(1), 120–130. https://doi.org/10.34312/euler.v10i1.14744

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning (Vol. 103). Springer New York. https://doi.org/10.1007/978-1-4614-7138-7

Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. Springer New York. https://doi.org/10.1007/978-1-4614-6849-3

Kvamme, H., Borgan, Ø., & Scheel, I. (2019). Time-to-Event Prediction with Neural Networks and Cox Regression. In Journal of Machine Learning Research (Vol. 20). http://jmlr.org/papers/v20/18-424.html.

Mittal, S., Madigan, D., Burd, R. S., & Suchard, M. A. (2014). High-dimensional, massive sample-size Cox proportional hazards regression for survival analysis. Biostatistics, 15(2), 207–221. https://doi.org/10.1093/biostatistics/kxt043

Purnami, S. W., & Pertiwi, I. N. (2020). Regresi Cox Proportional Hazard Untuk Analisis Survival Pasien Kanker Otak di C-Tech Labs Edwar Technology Tangerang. Inferensi, 3(2), 65. https://doi.org/10.12962/j27213862.v3i2.7727

Schober, P., & Vetter, T. R. (2018). Survival analysis and interpretation of time-to-event data: The tortoise and the hare. Anesthesia and Analgesia, 127(3), 792–798. https://doi.org/10.1213/ANE.0000000000003653

Smith, L. N. (2015). Cyclical Learning Rates for Training Neural Networks. http://arxiv.org/abs/1506.01186

Sullivan, L. (2016). Survival Analysis. Boston University School of Public Health.

Tarkhan, A., & Simon, N. (2020). BigSurvSGD: Big Survival Data Analysis via Stochastic Gradient Descent.

Turkson, A. J., Ayiah-Mensah, F., & Nimoh, V. (2021). Handling Censoring and Censored Data in Survival Analysis: A Standalone Systematic Literature Review. In International Journal of Mathematics and Mathematical Sciences (Vol. 2021). Hindawi Limited. https://doi.org/10.1155/2021/9307475

Uno, H., Cai, T., Pencina, M. J., D'Agostino, R. B., & Wei, L. J. (2011). On the C-statistics for evaluating overall adequacy of risk prediction procedures with censored survival data. Statistics in Medicine, 30(10), 1105–1117. https://doi.org/10.1002/sim.4154

Brownlee Jason. (2020, August 14). What is the Difference Between Test and Validation Datasets? Machine Learning Process.