BAB I
PENDAHULUAN

1.1 Latar Belakang

Verifikasi merupakan proses penting dalam menilai kualitas suatu prediksi, peringatan dini, atau ramalan (*forecast*). Proses ini membandingkan hasil prediksi dengan data observasi untuk mengevaluasi kesesuaian antara model dan realitas. Verifikasi tidak hanya bertujuan untuk memastikan akurasi prediksi tetapi juga untuk mengidentifikasi kelemahan model sehingga dapat ditingkatkan. Secara umum, verifikasi dapat dilakukan secara kualitatif untuk melihat kesesuaian pola prediksi-observasi, atau secara kuantitatif menggunakan metode statistik untuk mengukur tingkat akurasi dan kesalahan prediksi (Febrianti, 2021).

Salah satu metrik verifikasi yang efektif untuk menguji kemampuan prediktif suatu model adalah *Peirce Skill Score* (PSS). PSS merupakan indikator yang mengukur keahlian prediksi dalam membedakan antara kategori benar dan salah, tanpa terpengaruh oleh bias atau ketidakseimbangan data. Keunggulan PSS dibandingkan metrik lain (seperti *accuracy* atau *precision*) adalah kemampuannya memberikan evaluasi yang lebih stabil, terutama dalam kasus klasifikasi multi-kategori (Febrianti, 2021); (Martha & Herwindiati, 2024);

Dalam konteks klasifikasi tiga kategori, evaluasi menggunakan *confusion matrix* 3×3 menjadi penting karena mampu merepresentasikan hubungan prediksi-observasi secara lebih rinci dibandingkan matriks biner (2×2). Namun, perhitungan PSS berbasis *confusion matrix* 3×3 masih belum banyak dieksplorasi secara komputasional, padahal kebutuhan akan alat verifikasi yang akurat semakin meningkat seiring perkembangan model prediktif yang kompleks (Febrianti, 2021) ;(Maher & Sherwood, 2016).

Oleh karena itu, penelitian ini berfokus untuk memverifikasi atau menguji suatu prediksi dengan dilakukan perhitungan nilai akurasi prediksi menggunakan *Peirce Skill Score* (PSS) berbasis *confusion matrix 3×3*. Penelitian ini membangun sistem komputasi berbasis MATLAB untuk verifikasi prediksi melalui perhitungan PSS dari *confusion matrix* 3×3, dengan menggunakan matriks hasil klasifikasi curah hujan sebagai data uji dan ditampilkan dalam GUI MATLAB. Program ini dirancang sebagai alat verifikasi kuantitatif yang dapat diaplikasikan dalam berbagai bidang, mulai dari meteorologi hingga *machine learning*, dengan tujuan memberikan metode evaluasi prediksi yang lebih andal dan terstandarisasi. Hasil penelitian diharapkan dapat menjadi referensi bagi pengembangan teknik verifikasi prediksi, khususnya dalam penggunaan PSS untuk klasifikasi multi-kategori.

1.2 Tujuan dan Manfaat

1.2.1 Tujuan

1. Memverifikasi suatu prediksi dengan melakukan perhitungan nilai akurasi prediksi menggunakan *Peirce Skill Score* (PSS) berbasis confusion matrix 3×3.
2. Membangun sistem komputasi berbasis MATLAB untuk melakukan verifikasi prediksi melalui perhitungan PSS dari *confusion matrix* 3×3, serta mengimplementasikan kedalam bentuk antarmuka grafis (GUI) yang interaktif.
3. Memberikan pemahaman yang lebih mendalam terhadap penerapan PSS berbasis *confusion matrix* 3×3 sebagai metode verifikasi prediksi.

1.2.2 Manfaat

Penelitian ini bermanfaat sebagai referensi dalam penerapan *Peirce Skill Score* (PSS) untuk verifikasi prediksi multikategori, serta dapat mempermudah evaluasi performa suatu model klasifikasi secara praktis dan terukur.

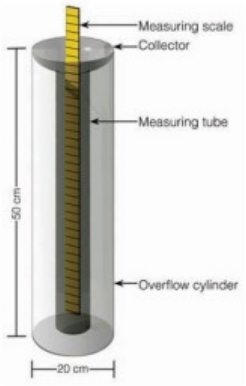
1.3 Landasan Teori

1.3.1 Curah Hujan

Curah hujan juga dapat didefinisikan sebagai jumlah air hujan yang terkumpul di daerah datar tanpa menguap, meresap, atau mengalir. Pada umumnya, curah hujan diukur menggunakan satuan per milimeter. Hal tersebut menunjukan bahwa per satu milimeter curah hujan berarti dalam luasan satu meter persegi pada tempat yang datar tertampung air setinggi satu milimeter atau tertampung air setinggi 1 liter (Alviriza Ramadhan dkk., 2024).

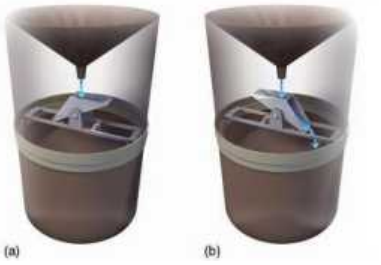
Setiap alat yang dapat mengumpulkan dan mengukur curah hujan disebut penakar hujan (*rain gauge*). Terdapat berbagai alat penagkar hujan diantaranya adalah penakar hujan standar dan penakar hujan ember jungkit. Penakar hujan standar terdiri dari corong penampung berbentuk kerucut yang terhubung ke tabung pengukur panjang (Gambar 1). Luas penampang corong ini 10 kali lebih besar dibandingkan dengan tabung pengukurnya. Dengan demikian, air hujan yang masuk ke corong akan diperbesar sepuluh kali lipat di dalam tabung, memungkinkan pengukuran dengan tingkat ketelitian tinggi. Sebuah penggaris kayu yang telah dikalibrasi untuk menyesuaikan pembesaran vertikal dimasukkan ke dalam tabung lalu ditarik keluar. Bagian basah pada penggaris menunjukkan kedalaman air. Jadi, jika terdapat 10 inci air di dalam tabung, maka akan terukur sebagai 1 inci curah hujan. Pembesaran ini memungkinkan pengukuran curah hujan dengan ketelitian hingga satu per seratus inci (0,01 inci). Jumlah curah hujan yang kurang dari 0,01 inci disebut jejak hujan (*trace*). Tabung pengukur hanya dapat

rah hujan. Curah hujan yang melebihi kapasitas ini akan meluap ke
li mana air kelebihan ini disimpan dan dilindungi dari penguapan
ns & Henson, 2019).



Gambar 1. Komponen Alat Ukur Hujan Standar
Sumber: Ahrens & Henson, 2019

Alat lain yang digunakan untuk mengukur curah hujan adalah penakar hujan ember jungkit (*tipping bucket rain gauge*). Pada Gambar 2, terlihat bahwa alat ini memiliki corong penerima yang mengarah ke dua wadah logam kecil (ember) yang saling terhubung dan dipasang pada sebuah poros. Ember yang berada di bawah corong akan menampung air hujan. Ketika air yang terkumpul mencapai setara dengan satu per seratus inci curah hujan, berat air tersebut akan menyebabkan ember tersebut jungkit dan mengosongkan isinya. Saat ember pertama berputar pada poros, ember kedua segera berpindah ke bawah corong untuk menampung air. Ketika ember kedua penuh, ia juga akan jungkit dan mengosongkan airnya ke arah berlawanan pada poros, dan ember pertama kembali ke posisi semula di bawah corong. Setiap kali ember menjungkit, sebuah kontak listrik terjadi, menyebabkan pena mencatat satu tanda pada grafik pencatat jarak jauh. Penjumlahan dari seluruh tanda ini menunjukkan jumlah curah hujan untuk periode waktu tertentu. Salah satu kelemahan dari penakar hujan ember jungkit adalah bahwa setiap kali ember menjungkit, sebagian air hujan dapat hilang, sehingga pengukuran curah hujan cenderung lebih rendah dari jumlah sebenarnya, terutama saat hujan deras (Ahrens & Henson, 2019).



Gambar 2. Penakar hujan ember jungkit
Sumber: Ahrens & Henson, 2019

an dari jarak jauh juga dapat dilakukan dengan alat pengukur hujan gan alat ini, curah hujan ditampung dalam sebuah silinder dan ember. Ember tersebut diletakkan di atas sebuah platform

penimbang yang sensitif. Rangkaian roda gigi khusus mengubah berat akumulasi hujan atau salju menjadi satuan milimeter atau inci curah hujan. Jumlah curah hujan tersebut dicatat oleh pena pada kertas grafik yang melapisi drum yang digerakkan oleh jam. Dengan menggunakan peralatan elektronik khusus, informasi ini dapat dikirimkan dari alat pengukur hujan di daerah terpencil ke satelit atau stasiun di darat, sehingga memungkinkan penyediaan data curah hujan dari wilayah yang sebelumnya tidak dapat dijangkau (Ahrens & Henson, 2019).

1.3.2 *Confusion matrix*

Confusion matrix adalah tabel khusus yang digunakan dalam pembelajaran mesin untuk menggambarkan dan menilai kinerja suatu model klasifikasi. Misalnya, model seperti jaringan saraf tiruan diuji dengan data yang jawabannya sudah diketahui. *Confusion matrix* untuk masalah klasifikasi dengan n kelas berbentuk persegi, dengan n baris dan n kolom. Baris-barisnya mewakili kelas dari sampel aktual (*instance*), yaitu *input* dari *classifier*, sedangkan kolom-kolomnya mewakili kelas dari sampel yang diprediksi, yaitu *output* dari *classifier*. Sebaliknya juga bisa dilakukan baris untuk prediksi dan kolom untuk data asli.

Confusion matrix dapat digunakan baik untuk klasifikasi biner (dua kelas) maupun multi-kelas (lebih dari dua kelas). Perlu dicatat bahwa istilah "*matrix*" di sini tidak berhubungan dengan rumus-rumus dalam aljabar matriks. matrix hanya dipakai sebagai tabel informasi. Kata "*confusion*" digunakan karena tabel ini menunjukkan sejauh mana model "bingung", yaitu salah mengklasifikasikan satu kelas sebagai kelas lain. Konsep dasar ini pertama kali diperkenalkan pada tahun 1904 oleh ahli statistik asal Inggris, Karl Pearson (1857–1936) (Fahmy Amin, 2022, 2023).

Mulai dari situasi dasar dan relatif sederhana, yaitu *binary classifier* (klasifikasi biner), di mana hanya ada dua kelas (n = 2) dan digunakan *Confusion matrix* berukuran 2x2. Lihat Gambar 3. Misalnya, matriks dalam gambar tersebut merupakan contoh dari sebuah tes medis yang dilakukan pada sejumlah orang (pasien) untuk mendeteksi ada atau tidaknya suatu penyakit. Label 'positif' (+ve) dan 'negatif' (-ve) digunakan untuk menunjukkan dua kondisi yang berbeda ini, yang diperlakukan sebagai dua kelas dalam masalah klasifikasi. (Label lain seperti ‘1’ dan ‘0’, ‘ya’ dan ‘tidak’, atau ‘terjadi’ dan ‘tidak terjadi’ juga bisa digunakan.) Dengan sistem pelabelan seperti ini, perhatian kadang lebih difokuskan pada kelas positif, dan hasil klasifikasinya dianggap sebagai penentu utama dalam mengevaluasi kinerja model.

		Predicted	
		+ve	-ve
Actual	+ve	100	5
	-ve	10	90

ıbar 3. *Confution Matrix* untuk Klasifikasi Binner
Sumber: (Fahmy Amin, 2022)

a Gambar 3 menunjukkan bahwa:

- Label 'positif' berarti orang tersebut mengidap penyakit, sedangkan label 'negatif' berarti orang tersebut tidak mengidap penyakit.
- Sebanyak 205 orang (= 100 + 5 + 10 + 90) telah diuji.
- Dari 205 orang tersebut, model memprediksi sebagai 'positif' sebanyak 110 kali (= 100 + 10) dan sebagai 'negatif' sebanyak 95 kali (= 5 + 90), terlepas dari apakah prediksi tersebut benar atau tidak.
- Secara aktual, 105 orang (= 100 + 5) dalam data uji benar-benar mengidap penyakit, dan 100 orang (= 10 + 90) tidak mengidap penyakit.

Secara formal, perbandingan antara klasifikasi sebenarnya dengan klasifikasi hasil prediksi menghasilkan empat kemungkinan yang jelas, yaitu:

- Klasifikasi sebenarnya adalah positif dan hasil prediksi juga positif. Ini disebut *true positive* (TP), karena sampel positif berhasil dikenali dengan benar oleh model.
- Klasifikasi sebenarnya adalah negatif dan hasil prediksi juga negatif. Ini disebut *true negative* (TN), karena sampel negatif berhasil dikenali dengan benar oleh model.
- Klasifikasi sebenarnya adalah negatif, tetapi hasil prediksi adalah positif. Ini disebut *false positive* (FP), karena sampel negatif salah dikenali sebagai positif oleh model.
- Klasifikasi sebenarnya adalah positif, tetapi hasil prediksi adalah negatif. Ini disebut *false negative* (FN), karena sampel positif salah dikenali sebagai negatif oleh model.

Keempat hasil ini, dengan penjelasan di atas, biasanya difokuskan pada kelas positif, terutama jika kelas tersebut dianggap penting dan perlu mendapat perhatian lebih. Dalam konteks ini, kelas positif disebut sebagai sampel yang relevan, sementara kelas negatif dianggap tidak relevan.

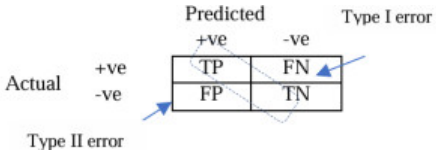
Hasil-hasil TP, TN, FP, dan FN sangat penting dan disebut sebagai *building blocks* (komponen dasar), karena digunakan untuk menyusun semua ukuran kinerja. Komponen-komponen dasar ini secara alami muncul sebagai elemen-elemen dalam *Confusion matrix*, seperti ditunjukkan pada Gambar 4 dan 5. Perlu dicatat bahwa hasil yang benar, yaitu TP dan TN, berada di dua sel diagonal matriks. Sementara itu, hasil yang salah, yaitu FP dan FN, berada di dua sel luar diagonal dan menunjukkan kesalahan; FN disebut sebagai *type I error*, sedangkan FP disebut sebagai *type II error*.

Dalam contoh kasus orang sakit dan sehat, FN mewakili orang yang sebenarnya sehat tetapi diklasifikasikan sebagai sakit, sedangkan FP mewakili orang yang sebenarnya sakit tetapi diklasifikasikan sebagai sehat. Kasus terakhir (*type II error*) biasanya lebih berbahaya dibandingkan kasus pertama (*type I error*). Kembali ke Gambar 3, komponen dasar terlihat sebagai berikut: TP = 100, TN = 90, FP = 10, FN = 5. Idealnya, FP dan FN bernilai nol, yang berarti model klasifikasi sempurna.

		Predicted	
		+ve	-ve
Actual	+ve	TP	FN
	-ve	FP	TN

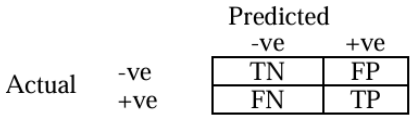
blocks dari kelas positif sebagai elemen-elemen dalam *Confusion*

matrix 2x2
Sumber: (Fahmy Amin, 2022)



Gambar 5. Hasil yang benar berada di sel diagonal dan hasil yang salah berada di sel luar diagonal
Sumber: (Fahmy Amin, 2022)

Namun, dalam praktiknya, ada tantangan dalam bagaimana cara meminimalkan FP dan FN (yaitu memaksimalkan TP dan TN). Perlu diingat bahwa komponen dasar tersebut semuanya berupa bilangan bulat positif (jumlah); tidak bisa berupa pecahan atau persentase. Perlu dicatat juga bahwa kelas positif dan negatif bisa saling ditukar, sehingga matriks kebingungan (*Confusion matrix*) akan tampak seperti pada Gambar 6. Jika dibandingkan dengan Gambar 4, kita akan melihat bahwa TP dan TN hanya saling bertukar tempat, begitu pula dengan FP dan FN.



Gambar 6. Pertukaran kelas positif dan negatif
Sumber: (Fahmy Amin, 2022)

Selain itu, kita dapat menuliskan:

- Jumlah sampel positif dalam himpunan uji:

$$N_+ = TP + FN \tag{1}$$

- Jumlah sampel negatif dalam himpunan uji:

$$N_- = FP + TN \tag{2}$$

- Jumlah total sampel yang diuji:

$$N = TP + FN + FP + TN = N_+ + N_- \tag{3}$$

- Jumlah sampel yang diprediksi sebagai positif:

$$P_+ = TP + FP \tag{4}$$

- Jumlah sampel yang diprediksi sebagai negatif:

$$P_- = FN + TN = N - P_+ \tag{5}$$

ang ditangani oleh suatu pengklasifikasi biner memiliki tingkat
npir sama, kedua set komponen dasarnya, beserta hubungan antar
dipelajari secara setara. Kelas individu diberi label secara
mberikan preferensi pada salah satu kelas. Gambar 7 menunjukkan

matriks kebingungan (*Confusion matrix*) untuk dua kelas yang dilabeli A dan B, di mana terlihat bahwa:

		Predicted A	Predicted B	
Actual	A	215	25	N_A=240
	B	40	190	N_B=230

Gambar 7. *Confusion matrix* dari pengklasifikasi biner dengan kelas A dan B
Sumber: (Fahmy Amin, 2022)

- Jumlah sampel kelas-A yang diuji:
  $N_A = 215 + 25 = 240$
- Jumlah sampel kelas-B yang diuji:
  $N_B = 40 + 190 = 230$
- Jumlah total sampel yang diuji:
  $N = N_A + N_B = 240 + 230 = 470$

Dalam hal ini, istilah 'positif' dan 'negatif' tidak muncul secara eksplisit, namun maknanya tetap tersirat. Jika yang diperhatikan adalah kelas A, maka dapat dipahami bahwa:

- Sampel kelas A yang diklasifikasikan dengan benar adalah TP_A (*true positive* untuk kelas A); $TP_A = 215$
- Sampel kelas B yang diklasifikasikan dengan benar adalah TNA (*true negative* untuk kelas A); $TN_A = 190$
- Sampel kelas B yang salah diklasifikasikan sebagai kelas A adalah FP_A (*false positive* untuk kelas A); $FP_A = 40$
- Sampel kelas A yang salah diklasifikasikan sebagai kelas B adalah FN_A *(false negative* untuk kelas A); $FN_A = 25$

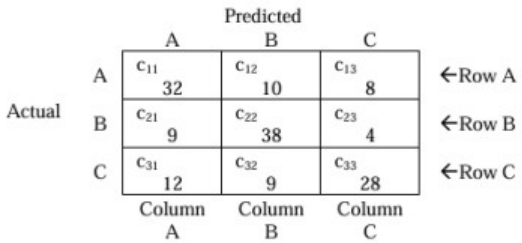
Sementara jika yang diperhatikan adalah kelas B, maka dapat dipahami bahwa:

- Sampel kelas B yang diklasifikasikan dengan benar adalah TP_B (*true positive* untuk kelas B); $TP_B = 190$
- Sampel kelas A yang diklasifikasikan dengan benar adalah TN_B (*true negative* untuk kelas B); $TN_B = 215$
- Sampel kelas A yang salah diklasifikasikan sebagai kelas B adalah FP_B (*false positive* untuk kelas B); $FP_B = 25$
- Sampel kelas B yang salah diklasifikasikan sebagai kelas A adalah FN_B (*false negative* untuk kelas B); $FN_B = 40$

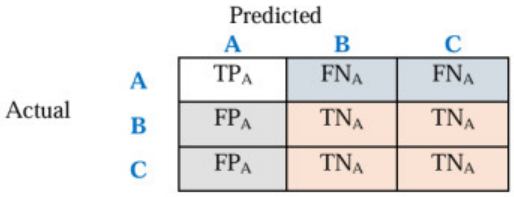
Konsep *Confusion matrix* yang digunakan pada klasifikasi biner dapat diperluas untuk
fikasi multi-kelas. Sebagai langkah awal dalam proses generalisasi
ga kelas (n = 3) digunakan. *Confusion matrix* terdiri atas tiga baris
ti ditunjukkan pada Gambar 8. Kelas-kelas diberi label A, B, dan C.
ks yang berada pada perpotongan baris ke-i dan kolom ke-j diberi
alam sel yang ditampilkan pada Gambar 8 merupakan contoh hasil
tal N = 150 sampel yang diuji dalam kesembilan sel tersebut.

Untuk kelas A, definisinya sebagai berikut:

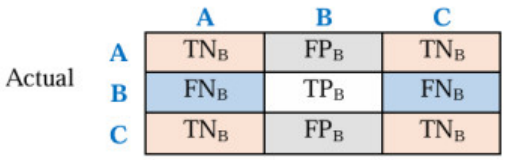
- TP_A: Sampel dari kelas A yang diklasifikasikan dengan benar sebagai kelas A. Nilainya adalah sel c_{11}, yaitu perpotongan antara baris A dan kolom A (lihat Gambar 8).
- TN_A: Sampel yang bukan berasal dari kelas A (yakni kelas B atau C) yang diklasifikasikan dengan benar atau salah sebagai bukan kelas A. Nilainya merupakan jumlah dari empat sel: c_{22}, c_{23}, c_{32}, dan c_{33}, yaitu bagian matriks yang tersisa setelah baris dan kolom A dihapus.
- FP_A: Sampel yang bukan berasal dari kelas A, namun salah diklasifikasikan sebagai kelas A. Nilainya adalah jumlah dari dua sel: c_{21} dan c_{31}, yaitu bagian dari kolom A selain sel c_{11}.
- FN_A: Sampel dari kelas A yang salah diklasifikasikan sebagai bukan kelas A. Nilainya adalah jumlah dari dua sel: c_{12} dan c_{13}, yaitu bagian dari baris A selain sel c_{11}.



Gambar 8. Confusion matrix untuk klasifikasi 3 kelas
Sumber: (Fahmy Amin, 2023)



Gambar 9. Komponen dasar untuk kelas A dalam klasifikasi 3 kelas
Sumber: (Fahmy Amin, 2023)



Komponen dasar untuk kelas B dalam klasifikasi 3 kelas
Sumber: (Fahmy Amin, 2023)

Actual	A	B	C
A	TN_C	TN_C	FP_C
B	TN_C	TN_C	FP_C
C	FN_C	FN_C	TP_C

Gambar 11. Komponen dasar untuk kelas C dalam klasifikasi 3 kelas
Sumber: (Fahmy Amin, 2023)

Gambar 9 menunjukkan, pada *Confusion matrix*, komponen dasar TP_A, TN_A, FP_A, dan FN_A untuk kelas A. Dengan cara yang sama, komponen dasar untuk kelas B dan C juga dapat didefinisikan. Lihat Gambar 10 dan 11. Untuk memudahkan, saat mempertimbangkan satu kelas, kelas tersebut dianggap sebagai kelas positif, sementara dua kelas lainnya dianggap sebagai kelas negatif.

Secara numerik, untuk kelas A:
$TP_A = c_{11} = 32$
$TN_A = c_{22} + c_{23} + c_{32} + c_{33} = 38 + 4 + 9 + 28 = 79$
$FP_A = c_{21} + c_{31} = 9 + 12 = 21$
$FN_A = c_{12} + c_{13} = 10 + 8 = 18$

Untuk kelas B:
$TP_B = c_{22} = 38$
$TN_B = c_{11} + c_{13} + c_{31} + c_{33} = 32 + 8 + 12 + 28 = 80$
$FP_B = c_{12} + c_{32} = 10 + 9 = 19$
$FN_B = c_{21} + c_{23} = 9 + 4 = 13$

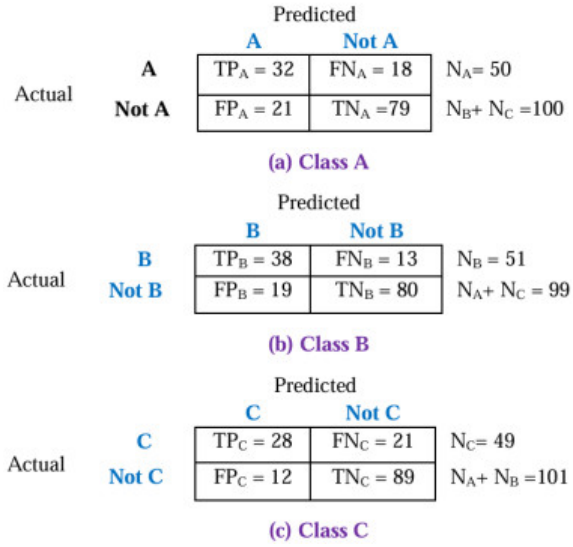
Untuk kelas C:
$TPC = c_{33} = 28$
$TNC = c_{11} + c_{12} + c_{21} + c_{22} = 32 + 10 + 9 + 38 = 89$
$FPC = c_{13} + c_{23} = 8 + 4 = 12$
$FNC = c_{31} + c_{32} = 12 + 9 = 21$

Perlu diperhatikan bahwa $TP_A = 32$, $TP_B = 38$, dan $TP_C = 28$ merupakan elemen diagonal dari *Confusion matrix* pada Gambar 8. Komponen dasar untuk ketiga kelas A, B, dan C dirangkum pada Gambar 12. Jumlah TP + TN + FP + FN untuk masing-masing kelas, seperti yang diharapkan, sama dengan jumlah total sampel, yaitu N = 150. Berdasarkan Gambar 8, diketahui bahwa $N_A = 50$, $N_B = 51$, dan $N_C = 49$.

Hal yang menarik adalah bahwa *Confusion matrix* berukuran 3x3 dapat diuraikan menjadi tiga *Confusion matrix* berukuran 2x2 (yang serupa dengan yang digunakan dalam klasifikasi biner). Dengan menggunakan nilai-nilai dari komponen dasar untuk g ditunjukkan pada Gambar 12, *Confusion matrix* 3x3 pada Gambar ga komponen *Confusion matrix* 2x2 untuk masing-masing kelas A, g ditampilkan pada Gambar 13.

	TP	TN	FP	FN	
Class A	32	79	21	18	N = 150
Class B	38	80	19	13	N = 150
Class C	28	89	12	21	N = 150

Gambar 12. Komponen dasar untuk setiap kelas dalam klasifikasi 3 kelas pada Gambar 8
Sumber: (Fahmy Amin, 2023)



Gambar 13. Tiga komponen *Confusion matrix* 2x2 untuk *Confusion matrix* 3x3 pada Gambar 8
Sumber: (Fahmy Amin, 2023)

Nilai sel dari tiga *Confusion matrix* 2x2 pada Gambar 13 memberikan informasi yang sama dengan tiga baris tabel pada Gambar 12, yaitu elemen dasar dari masing-masing kelas. Dalam hal ini, dapat disimpulkan bahwa masalah klasifikasi 3 kelas diubah menjadi tiga masalah klasifikasi biner.

Dalam penelitian sebelumnya, telah dibahas secara rinci sekelompok ukuran kinerja untuk model klasifikasi biner yang didasarkan pada *Confusion matrix* yang relevan. Ukuran-ukuran tersebut diperluas di sini untuk klasifikasi tiga kelas. Definisi dan implikasinya, seperti yang akan terlihat, pada dasarnya tetap sama, dengan
ɔahwa *Confusion matrix* berbentuk 3x3 dan elemen dasar untuk
ditentukan sebagaimana dijelaskan sebelumnya.

ı C, *Confusion matrix* memiliki bentuk umum seperti pada Gambar
ɔonen *Confusion matrix* 2x2 seperti pada Gambar 9. Simbol E_{AB}
ampel kelas A yang salah diklasifikasikan sebagai kelas B, dan

		Predicted A	Predicted B	Predicted C	
Actual	A	TP_A	E_{AB}	E_{AC}	N_A
	B	E_{BA}	TP_B	E_{BC}	N_B
	C	E_{CA}	E_{CB}	TP_C	N_C

Gambar 14. Bentuk umum dari *Confusion matrix* 3x3
Sumber: (Fahmy Amin, 2023)

		Predicted A	Predicted Not A
Actual	A	TP_A	$FN_A = E_{AB} + E_{AC}$
	Not A	$FP_A = E_{BA} + E_{CA}$	$TN_A = TP_B + E_{BC} + E_{CB} + TP_C$

(a) Class A

		Predicted B	Predicted Not B
Actual	B	TP_B	$FN_B = E_{BA} + E_{BC}$
	Not B	$FP_B = E_{AB} + E_{CB}$	$TN_B = TP_A + E_{AC} + E_{CA} + TP_C$

(b) Class B

		Predicted C	Predicted Not C
Actual	C	TP_C	$FN_C = E_{CA} + E_{CB}$
	Not C	$FP_C = E_{AC} + E_{BC}$	$TN_C = TP_A + E_{AB} + E_{BA} + TP_B$

(a) Class C

Gambar 15. Tiga *Confusion matrix* 2x2 untuk *Confusion matrix* 3x3 pada Gambar 14
Sumber: (Fahmy Amin, 2023)

Jumlah total sampel yang diuji adalah

$$N_A + N_B + N_C = P_A + P_B + P_C = N \quad (6)$$

Jumlah sampel aktual untuk setiap kelas adalah:

$$N_A = TP_A + E_{AB} + E_{AC} = TP_A + FN_A \quad (7a)$$
$$N_B = TP_B + E_{BA} + E_{BC} = TP_B + FN_B \quad (7b)$$
$$N_C = TP_C + E_{CA} + E_{CB} = TP_C + FP_C \quad (7c)$$

Jumlah sampel yang diprediksi untuk setiap kelas adalah:

$$P_A = TP_A + E_{BA} + E_{CA} = TP_A + FP_A \quad (8a)$$
$$P_B = TP_B + E_{AB} + E_{CB} = TP_B + FP_B \quad (8b)$$
$$P_C = TP_C + E_{AC} + E_{BC} = TP_C + FP_C \quad (8c)$$

iraan dengan Tabel Kontingensi

ebih mudah dipahami dalam konteks prakiraan nonprobabilistik
et, yang berarti prakiraan tersebut hanya menyatakan bahwa satu

hasil tertentu akan terjadi, tanpa menyebutkan kemungkinan atau ketidakpastian. Prediktan diskret adalah variabel yang hanya dapat memiliki salah satu dari sejumlah nilai yang terbatas, berbeda dengan prediktand kontinu yang bisa mengambil nilai apa saja dalam rentang yang lebih luas. Verifikasi prakiraan jenis ini telah dilakukan sejak abad ke-19, meskipun dengan penggunaan berbagai istilah yang kadang membingungkan. Sebagai contoh, istilah "kategori" sering digunakan untuk merujuk pada prakiraan nonprobabilistik, namun kini istilah ini digantikan dengan "nonprobabilistik" dan "diskret" untuk menghindari kebingungannya (Wilks, 2019).

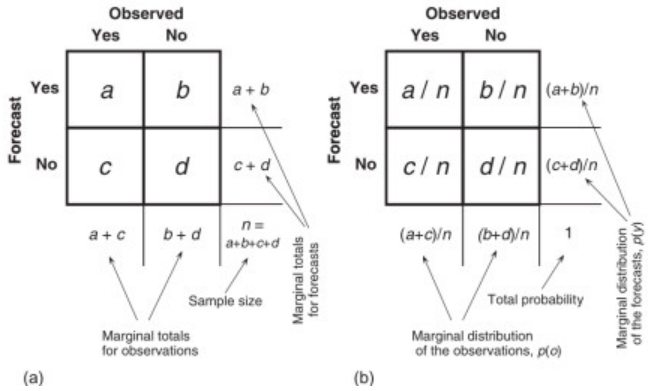
Dalam verifikasi prakiraan cuaca, sering kali ada hubungan langsung antara jenis prakiraan dan jenis kejadian yang diamati. Misalnya, jika kita hanya memprakirakan apakah hujan akan terjadi atau tidak, maka ada dua kemungkinan prakiraan: “ya” (akan hujan) dan “tidak” (tidak akan hujan). Begitu juga, hasil observasi (kenyataan) juga punya dua kemungkinan: “ya” (hujan benar-benar terjadi) dan “tidak” (hujan tidak terjadi). Jadi, baik prakiraan maupun observasi memiliki dua kategori inilah yang disebut sebagai kasus 2 × 2.

Data prakiraan dan hasil observasinya bisa disusun dalam bentuk tabel yang disebut tabel kontingensi (*contingency tabel*). Tabel ini mencatat berapa kali setiap kombinasi terjadi, misalnya berapa kali hujan diprakirakan dan benar-benar terjadi, berapa kali diprakirakan hujan tapi tidak hujan, dan seterusnya. Jika kita mengubah angka-angka ini menjadi bentuk persen (frekuensi relatif), maka kita mendapatkan gambaran seberapa sering tiap kombinasi terjadi, yang disebut distribusi gabungan.

Contohnya, jika hujan diprakirakan dan benar-benar terjadi sebanyak a kali dari total n kejadian, maka kita sebut ini sebagai *hit*, dan nilainya adalah a/n. Jika hujan diprakirakan tetapi tidak terjadi sebanyak b kali, ini disebut *false alarm*, nilainya b/n. Jika hujan terjadi tanpa diprakirakan sebanyak c kali, ini disebut *miss*, nilainya c/n. Terakhir, jika hujan tidak diprakirakan dan memang tidak terjadi sebanyak d kali, ini disebut *correct rejection* atau benar-negatif, nilainya d/n.

Biasanya, tabel ini juga dilengkapi dengan jumlah total pada baris dan kolom, yang menunjukkan seberapa sering prakiraan “ya” atau “tidak” muncul, serta seberapa sering kejadian sebenarnya “ya” atau “tidak”. Jumlah ini membantu memberikan gambaran menyeluruh tentang performa prakiraan.

Gambar 16 di bawah memperlihatkan hubungan antara jumlah pasangan prakiraan dan kejadian (ditunjukkan dengan huruf a–d) dalam situasi verifikasi nonprobabilistik dikotomis seperti yang ditampilkan dalam tabel kontingensi 2×2 (huruf tebal, panel a), dan distribusi gabungan yang sesuai antara prakiraan dan observasi (huruf tebal, panel
menunjukkan total marginal, yang menunjukkan seberapa sering
lua kejadian diprakirakan dan diamati dalam bentuk absolut; serta
ri observasi dan prakiraan, yang menyajikan informasi yang sama
ısi relatif (Wilks, 2019).



Gambar 16. Tabel Kontingensi dan Distribusi Probabilitas untuk Verifikasi Prakiraan Ya/Tidak
Sumber: Wilks, 2019

Singkatnya, gambar ini menjelaskan bagaimana data prakiraan dan observasi disusun dan diubah dari bentuk hitungan (absolut) menjadi bentuk probabilitas (relatif), yang merupakan dasar untuk menghitung berbagai skor keterampilan prakiraan seperti akurasi, sensitivitas dan PSS.

Meskipun tabel kontingensi 2×2 digunakan dalam prakiraan paling sederhana, tabel ini memuat informasi berdimensi tiga, sehingga tidak cukup dijelaskan hanya dengan satu atau dua statistik. Berbagai atribut skalar telah dikembangkan untuk menggambarkan kinerja prakiraan yaitu:

A. *Accuracy*

Akurasi mengacu pada kesesuaian rata-rata antara prakiraan individu dan kejadian yang diprediksi. Dalam konteks tabel kontingensi 2×2, ini menggambarkan seberapa baik prakiraan tersebut mencocokkan hasil yang terjadi, baik dalam hal kejadian yang diprakirakan benar maupun kejadian yang tidak diprakirakan. Akurasi memberikan gambaran umum tentang kualitas prakiraan, tetapi bisa jadi kurang tepat ketika kejadian yang diprakirakan jarang terjadi (Wilks, 2019).

Threat Score (TS) atau *Critical Success Index* (CSI) adalah alternatif yang lebih berguna untuk situasi di mana kejadian yang diprakirakan jarang terjadi. TS mengukur seberapa baik prakiraan dalam memprediksi kejadian yang terjadi tanpa menghitung prakiraan "tidak terjadi" yang benar. TS lebih fokus pada kejadian yang diprakirakan benar (*hit*) dan mengabaikan prakiraan yang benar untuk kejadian yang tidak terjadi, memberikan gambaran yang lebih jelas tentang kinerja prakiraan dalam situasi yang tidak seimbang antara kejadian dan ketidakterjadinya kejadian(Wilks, 2019). Nilainya berada di antara 0 nenunjukkan nilai yang baik (Reddy dkk., 2021).

$$\mathrm{TS} = \mathrm{CSI} = \frac{\mathrm{a}}{\mathrm{a} + \mathrm{b} + \mathrm{c}} \tag{9}$$

ur perbandingan antara probabilitas kejadian yang diprakirakan bilitas kesalahan prakiraan (*false alarm*). Dalam konteks verifikasi o membantu menilai seberapa baik prakiraan tersebut dapat

memisahkan antara kejadian yang terjadi dan yang tidak terjadi. Rasio ini sangat berguna untuk memahami sejauh mana prakiraan memprediksi kejadian dengan benar dibandingkan dengan kesalahan yang terjadi, memberikan wawasan lebih lanjut ke dalam kualitas diskriminasi prakiraan (Wilks, 2019).

$$\theta = \frac{ad}{bc} \tag{10}$$

B. *Bias*

Bias, atau perbandingan antara prakiraan rata-rata dengan pengamatan rata-rata, biasanya disajikan sebagai rasio untuk verifikasi tabel kontingensi. Dalam konteks tabel 2×2 pada Gambar 16, rasio bias adalah

$$B = \frac{a+b}{a+c} \tag{11}$$

Bias pada dasarnya adalah rasio antara jumlah prakiraan "ya" dengan jumlah pengamatan "ya". Prakiraan yang tidak bias menunjukkan B = 1, yang menunjukkan bahwa kejadian diprakirakan sebanyak jumlah kejadian yang teramati. Perlu dicatat bahwa bias tidak memberikan informasi tentang kesesuaian antara prakiraan dan pengamatan individu dari kejadian tersebut pada kesempatan tertentu, sehingga persamaan 11 bukanlah ukuran akurasi. Bias yang lebih besar dari satu menunjukkan bahwa kejadian diprakirakan lebih sering daripada yang teramati, yang disebut dengan *overforecasting*. Sebaliknya, bias yang kurang dari satu menunjukkan bahwa kejadian diprakirakan lebih jarang daripada yang teramati, atau disebut *underforecast (Wilks, 2019)*.

C. *Reliability* dan *Resolution*

Keandalan (*reliability*) dan resolusi (*resolution*) adalah dua konsep penting dalam menilai kinerja prakiraan yang menggunakan tabel kontingensi 2×2. *reliability* mengukur sejauh mana prakiraan "ya" atau "tidak" sesuai dengan kejadian yang teramati. Ini diukur dengan statistik seperti *False Alarm Ratio* (FAR). FAR menunjukkan seberapa sering prakiraan "ya" ternyata salah, yaitu perbandingan antara jumlah prakiraan yang salah (b) dan total prakiraan "ya" (a + b). Rasio FAR dihitung dengan rumus

$$FAR = \frac{b}{a+b} \tag{12}$$

di mana nilai FAR yang lebih kecil dianggap lebih baik, dengan nilai terbaik adalah nol (yang berarti tidak ada alarm palsu) dan terburuk adalah satu (yang berarti semua prakiraan "ya" salah).

resolution mengukur kemampuan prakiraan untuk membedakan kejadian yang berbeda.
edaan antara kejadian yang diprakirakan dan yang tidak, semakin
eandalan dan resolusi sering kali saling terkait, karena keduanya
berapa baik prakiraan menggambarkan distribusi kejadian yang
eluruhan, FAR memberikan gambaran tentang kualitas prakiraan
esalahan, terutama dalam prakiraan yang langka atau tidak sering

D. *Discriminant*

Diskriminasi dalam konteks verifikasi prakiraan merujuk pada kemampuan prakiraan untuk membedakan kejadian yang benar-benar terjadi (positif) dan yang tidak terjadi (negatif). Dua statistik utama digunakan untuk mengukur diskriminasi dalam tabel kontingensi 2×2:

Hit Rate adalah rasio antara prakiraan yang benar (kejadiannya sesuai dengan prakiraan) dengan jumlah kejadian yang sebenarnya terjadi. *Hit Rate* menunjukkan probabilitas deteksi, yaitu seberapa sering prakiraan yang benar terjadi ketika kejadian tersebut memang terjadi. Dalam statistik medis, ini dikenal sebagai fraksi positif yang benar atau sensitivitas. *Hit Rate* dihitung dengan rumus:

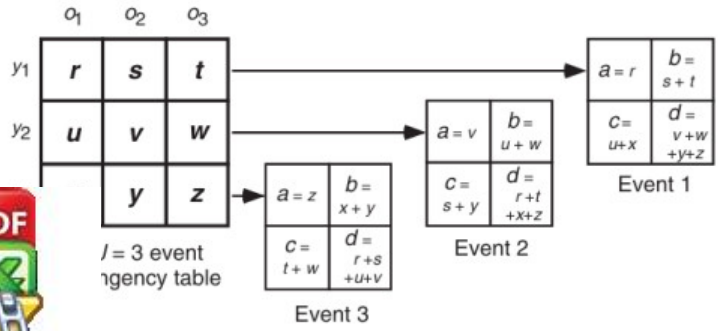
$$H = \frac{a}{a + c} \tag{13}$$

False Alarm Rate adalah rasio antara alarm palsu (prakiraan yang salah bahwa kejadian akan terjadi) dengan total jumlah kejadian yang tidak terjadi. *False Alrm Rate* mengukur probabilitas deteksi palsu, yaitu seberapa sering prakiraan yang salah muncul ketika kejadian tersebut tidak terjadi. *False Alarm Rate* dihitung dengan rumus:

$$F = \frac{b}{b + d} \tag{14}$$

Secara keseluruhan, *Hit Rate* dan *False Alarm Rate* bekerja bersama untuk memberi gambaran yang lebih jelas tentang kinerja prakiraan, terutama dalam hal membedakan antara kejadian yang benar-benar terjadi dan yang tidak terjadi (Wilks, 2019).

Prakiraan nonprobabilistik untuk prediktand diskrit tidak terbatas pada format 2×2, meskipun format ini adalah yang paling sering ditemui dan mudah dipahami. Dalam beberapa situasi, terutama jika ada lebih dari dua kejadian yang saling eksklusif dan kolektif, penggunaan tabel kontingensi lebih besar, seperti 3×3, diperlukan, terlihat pada Gambar 17. Pada tabel kontingensi 3×3, terdapat sembilan kemungkinan hasil untuk pasangan prakiraan dan kejadian. Proporsi benar (PC) dalam kasus ini dihitung dengan rumus PC = (r + v + z) / n, di mana r, v, dan z adalah jumlah kejadian yang diprakirakan dengan benar, dan n adalah total jumlah prakiraan. Rumus ini menunjukkan jumlah prakiraan yang benar dibagi dengan jumlah total prakiraan.



/ = 3 event
ngency table

el kontingensi untuk situasi verifikasi prakiraan nonprobabilistik dengan I = J = 3.
Sumber: Wilks, 2019

Untuk ukuran akurasi lainnya yaitu pada persamaan 9 sampai 14, yang berlaku untuk situasi dikotomis, perlu dilakukan penguraian tabel kontingensi 3×3 menjadi beberapa tabel kontingensi 2×2. Setiap tabel 2×2 dibangun dengan membandingkan kejadian yang diprakirakan ("kejadiannya") dengan kejadian komplementernya, yaitu "bukan kejadian yang diprakirakan." Sebagai contoh, pada tabel kontingensi 2×2 untuk Kejadian 1, Kejadian 2 dan Kejadian 3 digabungkan sebagai “bukan Kejadian 1.” Dalam hal ini, bias untuk prakiraan Kejadian 1 dihitung dengan rumus $B_1 = (r + s + t) / (r + u + x)$, bias untuk prakiraan Kejadian 2 dihitung dengan rumus $B_2 = (u + v + w) / (s + v + y)$, dan bias untuk prakiraan Kejadian 3 dihitung dengan rumus $B_3 = (x + y + z) / (t + w + z)$. Dengan demikian, rumus bias ini memberikan gambaran tentang seberapa sering kejadian diprakirakan lebih atau kurang dari frekuensi aktualnya dalam data (Wilks, 2019).

1.3.4 *Peirce Skill Score*

Peirce Skill Score (PSS) adalah salah satu skor keterampilan yang populer digunakan untuk mengevaluasi akurasi prakiraan dalam bentuk tabel kontingensi dua arah (2x2). *Peirce Skill Score* (PSS) pertama kali diterbitkan pada tahun 1884 dan sejak saat itu telah ditemukan kembali dengan nama *Kuipers Performance Index* dan *True Skill Statistic* (TSS), sebagaimana dibahas dalam (Stephenson, 2000).

Peirce Skill Score (PSS) menggunakan acuan berupa prakiraan acak yang diasumsikan tidak bias sebagai nilai pada bagian penyebut. Artinya, prakiraan acak tersebut memiliki distribusi marginal yang sama dengan data klimatologis atau distribusi historis dari kejadian yang diamati. Dengan demikian, probabilitas kejadian yang diprakirakan sama dengan probabilitas kejadian yang diamati, yaitu $p(y_1) = p(o_1)$ dan $p(y_2) = p(o_2)$. *Peirce Skill Score* dihitung sebagai berikut (Wilks, 2019).

$$\mathrm{PSS}_{2x2} = \frac{(a + d)/n - [(a + b)(a + c) + (b + d)(c + d)]/n^2}{1 - [(a + c)^2 + (b + d)^2]/n^2}$$
$$\mathrm{PSS}_{2x2} = \frac{ad - bc}{(a + c)(b + d)} \quad (15)$$

Peirce Skill Score (PSS) juga dapat dipahami sebagai selisih antara dua probabilitas kondisional dalam pemisahan faktor *likelihood-base rate* dari distribusi gabungan, yaitu antara *Hit Rate* (tingkat keberhasilan mendeteksi kejadian yang benar-benar terjadi) dan *False Alarm Rate* (tingkat kesalahan memprediksi kejadian yang sebenarnya tidak terjadi). Dengan kata lain, PSS = H – F.

Skor PSS akan bernilai satu jika prakiraan sangat akurat (karena nilai b dan c sama dengan nol, atau dengan kata lain H = 1 dan F = 0). Jika prakiraan dilakukan secara
n bernilai nol (karena H = F). Sementara itu, jika prakiraan lebih
aan acak, maka skor akan bernilai negatif. Prakiraan yang selalu
egori saja, seperti selalu memilih y_1 atau y_2, juga akan mendapat

ontingensi 3x3 seperti ditunjukkan pada Gambar 20, maka *Peirce*
ebagai berikut.

$$\mathrm{PSS}_{3x3} = \frac{\sum_{i=1}^{I} p(y_i, o_i) - \sum_{i=1}^{I} p(y_i)p(o_i)}{1 - \sum_{j=1}^{J}\left[p(o_j)\right]^2} \quad (16)$$

Peirce Skill Score memberikan kontribusi yang lebih besar untuk prakiraan yang benar tergantung pada seberapa jarang atau sering kejadian tersebut. Artinya, jika suatu kejadian tergolong langka, maka prakiraan yang benar terhadap kejadian tersebut akan memberikan nilai tambah yang lebih besar pada skor. Dengan cara ini, PSS tidak memberikan penalti terhadap prakiraan untuk kejadian langka hanya karena kejadian tersebut memiliki probabilitas klimatologis yang rendah (Wilks, 2019).

Standard error dari *Peirce Skill Score* (ePSS) merupakan ukuran statistik yang digunakan untuk menggambarkan derajat ketidakpastian dari nilai PSS yang diperoleh berdasarkan data sampel. Nilai ini mencerminkan seberapa besar kemungkinan nilai PSS tersebut berfluktuasi apabila pengamatan dilakukan berulang kali dengan data yang berbeda namun berasal dari populasi yang sama. Untuk tabel *Confusion Matrix* 2x2 *Standard error* dari *Peirce Skill Score* (ePSS) dirumuskan sebagai berikut (Halide & Ridd, 2008);(Stephenson, 2000);(Woodcock, 1976).

$$\mathrm{ePSS}_{2x2} = \sqrt{\frac{n^2 - 4(a+c)(b+d) \times PSS^2}{4n(a+c)(b+d)}} \quad (17)$$

Untuk tabel *Confusion Matrix* 3x3, *Standard error* dari *Peirce Skill Score* (ePSS) dirumuskan sebagai berikut.

$$\mathrm{ePSS}_{3x3} = \sqrt{\sum_{i,j}\left(\frac{\partial \mathrm{PSS}}{\partial p_{ij}}\right)^2 . \frac{p_{ij}(1 - p_{ij})}{n}} \quad (18)$$

1.3.5 *Graphic User Interface* (GUI) MATLAB

Matlab (*Matrix Laboratory*) adalah sebuah program untuk analisis dan komputasi numerik serta merupakan suatu bahasa pemograman matematika lanjutan yang dibentuk dengan dasar pemikiran menggunakan sifat dan bentuk matrik. Pada awalnya, program ini merupakan interface untuk koleksi rutin numerik proyek LINPACK dan EISPACK dan dikembangkan menggunakan bahasa Fortran. Namun sekarang program ini merupakan produk komersial dari perusahaan Mathworks, Inc. yang dalam perkembangan selanjutnya dikembangkan menggunakan bahasa C++ dan *Assembler*. Matlab telah berkembang menjadi sebuah environment pemograman yang canggih dan berisi fungsi-fungsi *built-in* untuk melakukan tugas pengolahan sinyal, aljabar linier dan kalkulasi atlab juga berisi *toolbox* yang berisi fungsi-fungsi tambahan untuk ab juga bersifat *extensible*, dalam arti bahwa seorang pengguna baru untuk ditambahkan di *library* jika fungsi-fungsi *built-in* yang t melakukan tugas tertentu. Kemampuan pemograman yang alu sulit bila pembaca telah memiliki pengalaman dalam bahasa lain seperti C, Pascal atau Fortran. Matlab merupakan bahasa

pemograman tingkat tinggi berbasis pada matriks sering digunakan untuk teknik komputasi dan digunakan untuk menyelesaikan masalah-masalah yang melibatkan operasi matematika, elemen matrik, optimasi, aproksimasi dan lain-lain. Selain itu Matlab banyak digunakan untuk matematika dan komputasi, pengembangan dan algoritma, pemograman pemodelan, simulasi dan pembuatan prototipe, analisa data, eksplorasi dan visualisasi, analisa numerik dan statistik serta pengembangan aplikasi teknik (Laksono, 2017).

Graphical User Interface (GUI) adalah salah satu bentuk antarmuka manusia-komputer (*human-computer interface*) yang dirancang untuk memudahkan interaksi antara pengguna dan komputer. GUI muncul sebagai solusi terhadap masalah layar kosong pada komputer awal yaitu hanya menampilkan *command prompt* tanpa petunjuk atau tampilan visual apapun, di mana pengguna tidak memiliki petunjuk mengenai langkah selanjutnya. Secara umum, GUI merupakan tampilan visual yang menghubungkan pengguna dengan fungsi dan potensi sistem komputer. Antarmuka ini memungkinkan pengguna untuk mengoperasikan aplikasi dengan cara yang intuitif, sehingga mereka dapat lebih fokus pada tugas utama tanpa harus memahami detail teknis sistem. GUI biasanya memiliki karakteristik WIMP (*Windows*, *Icons*, *Menus*, *Pointers*) dan terdiri dari tiga komponen utama, yaitu (Bernard J. Jansen, 1998):

- *Windowing system*: membangun jendela, menu, dan dialog.
- *Imaging model*: menentukan font dan tampilan grafis.
- *Application Program Interface* (API): sarana untuk mengatur tampilan dan interaksi.

Guide atau GUI (*Graphical User Interface*) adalah salah satu komponen dari Matlab untuk membuat interface (*desain form*) proses penyelesaian persoalan matematika yang lebih efisien dan menarik. Tidak seperti m-file hanya bisa bermain di *Command Windows*. Di sini, *user* akan membuat form (lembar kerja) untuk masing-masing program aplikasi dengan menggunakan atribut yang sudah disediakan oleh Matlab (Retno Murniasih dkk., 2021).

BAB II
METODE PENELITIAN

2.1 Data Penelitian

Dalam penelitian ini, data utama yang digunakan berupa tabel *Confusion matrix* 3×3 yang diambil dari Tabel 9.3 dalam buku *Statistical Methods in the Atmospheric Sciences* edisi keempat karya Daniel S. Wilks (2019). Seperti yang terlihat pada Gambar tabel dibawah.

TABLE 9.3 Nonprobabilistic MOS Forecasts for Freezing Rain (y_1), Snow (y_2), and Rain (y_3), Conditional on Occurrence of Some Form of Precipitation, for the Eastern Region of the United States During Cool Seasons of 1983/1984 Through 1988/1989

Full 3 × 3 Contingency Table

	o_1	o_2	o_3
y_1	50	91	71
y_2	47	2364	170
y_3	54	205	3288

Freezing Rain

	o_1	Not o_1
y_1	50	162
Not y_1	101	6027

TS = 0.160
θ = 18.4
B = 1.40
FAR = 0.764
H = 0.331
F = 0.026

Snow

	o_2	Not o_2
y_2	2364	217
Not y_2	296	3463

TS = 0.822
θ = 127.5
B = 0.97
FAR = 0.084
H = 0.889
F = 0.059

Rain

	o_3	Not o_3
y_3	3288	259
Not y_3	241	2552

TS = 0.868
θ = 134.4
B = 1.01
FAR = 0.073
H = 0.932
F = 0.092

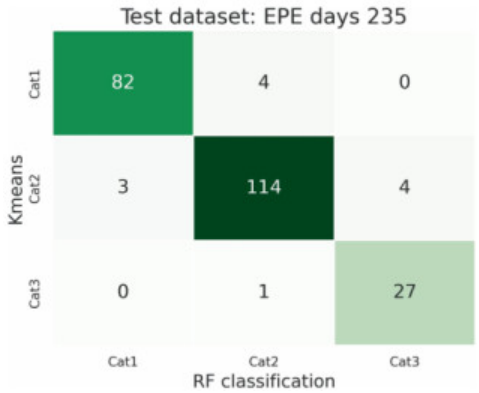
The verification data are presented as a 3 × 3 contingency table on the left, and then as three 2 × 2 contingency tables for each of the three precipitation types. Also shown are scalar attributes from Section 9.2.2 for each of the 2 × 2 tables. The sample size is n = 6340. Data are from Goldsmith (1990).

Gambar 18. Data utama tabel *Confusion matrix* 3x3
Sumber: Wilks, 2019

Tabel tersebut menyajikan hasil verifikasi prakiraan curah hujan dalam bentuk tiga kategori cuaca: hujan beku (*freezing rain*), salju (*snow*), dan hujan (*rain*). Data mencakup 6.340 kasus observasi selama musim dingin dalam periode lima tahun (1983–1989) di wilayah timur Amerika Serikat. Data verifikasi ditampilkan dalam dua format yaitu tabel kontingensi 3×3 penuh, yang menunjukkan jumlah prediksi dan observasi untuk ketiga jenis presipitasi (kategori y_1, y_2, y_3 terhadap observasi o_1, o_2, o_3) dan tiga tabel 2×2 turunan, masing-masing berfokus pada satu kategori sebagai kejadian target untuk memudahkan penghitungan metrik statistik yaitu TS (*Threat Score*), θ (*theta*), FAR (*False Alarm Rate*), B (*Bias*), H (*Hit Rate*), dan F (*False Alarm Ratio*). Data tabel *Confusion matrix* 3x3 tersebut dijadikan sebagai data utama dalam membuat program matlab untuk penentuan *Peirce Skill Score* (PSS) dan juga metrik tiap kategori. Selain itu nilai PSS yang diperoleh dari *Confusion matrix* 3x3 tersebut sebesar 0.8108.

ata *tabel Confusion matrix* 3x3 yang diambil dari dua studi kasus ifikasi model prediksi hujan menggunakan pendekatan *machine* bel *Confusion matrix* 3x3 diperoleh dari jurnal berjudul "*Improving* ation Extremes over Northern and Central Italy Using Machine k., 2024). Jurnal ini memaparkan kinerja model MaLCoX (*Machine licting Conditions for eXtreme precipitation*), yang menggunakan

algoritma *Random Forest* untuk mendeteksi dan mengklasifikasi kejadian hujan ekstrem (*Extreme Precipitation Events*/EPEs) di Italia bagian utara dan tengah. Tabel *Confusion matrix* 3x3 dalam jurnal ini menunjukkan akurasi model dalam membedakan tiga kategori EPE berdasarkan mekanisme meteorologis, yaitu *uplift orografis* (Cat1), *uplift frontal* (Cat2), dan konveksi termal (Cat3). Pengklasifikasian kategori kejadian presipitasi ekstrem EPEs didasarkan pada analisis pola fisik atmosfer menggunakan metode klastering *K-means* terhadap sejumlah parameter meteorologi (Grazzini dkk., 2020). Tabel *Confusion matrix* 3x3 tersebut terlihat pada Gambar 19.



Gambar 19. Data tabel *Confusion matrix* 3x3 Jurnal Grazzini dkk., 2020
Sumber: Grazzini dkk., 2020

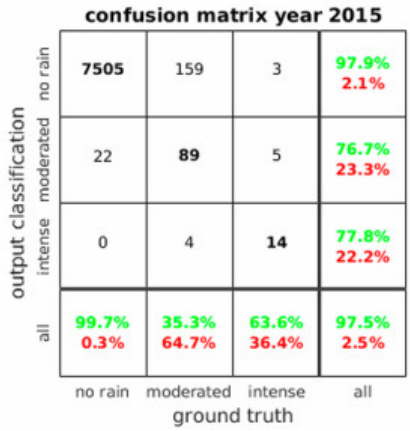
Kedua, data tabel *Confusion matrix* 3x3 yang diambil dari jurnal "*Neural Network Approach to Forecast Hourly Intense Rainfall Using GNSS Precipitable Water Vapor and Meteorological Sensors*" (Benevides dkk., 2019). Penelitian ini menggunakan model jaringan saraf NARX (*Nonlinear Autoregressive with Exogenous Inputs*) yang menggabungkan data precipitabel water vapor (PWV) dari GNSS serta data meteorologi dan satelit SEVIRI untuk memprediksi intensitas curah hujan per jam. Tabel *Confusion matrix* 3x3 dalam jurnal ini digunakan untuk mengevaluasi kemampuan model dalam mengklasifikasikan curah hujan ke dalam tiga kategori: tidak hujan, hujan sedang, dan hujan intens. pengklasifikasian kategori *precipitation class* (kelas curah hujan) dilakukan berdasarkan nilai ambang (*threshold*) dari intensitas curah hujan per jam. Batas 5 mm/jam digunakan sebagai ambang hujan intens karena dianggap cukup signifikan secara operasional dan karena data intensitas hujan ≥5 mm/jam hanya mencakup sekitar 0.5% dari total data (kejadian langka, sesuai untuk pengujian ekstrem).

Table 3. Details of the classification of precipitation data and the corresponding number of observations concerning the neural network data division.

ın Class	Rain Classification (mm/h)	Number of Samples (hour)		
		4 Years	1 Year	Total
in	0	31284	7527	38811
d rain	> 0 < 5	2184	252	2436
rain	>= 5	159	22	181

Klasifikasi kelas curah hujan jurnal Benevides dkk., 2019
Sumber: Benevides dkk., 2019

Data tabel *Confusion matrix* 3x3 tersebut terlihat pada Gambar 21.



Gambar 21. Data tabel *Confusion matrix* 3x3 Jurnal Benevides dkk., 2019
Sumber: Benevides dkk., 2019

Confusion matrix dari kedua jurnal tersebut dijadikan sumber data uji karena menggambarkan performa klasifikasi dari masing-masing model prediksi, sehingga dapat digunakan untuk analisis dan verifikasi kinerja prakiraan curah hujan dalam penelitian ini.

2.2 Prosedur Penelitian

Penelitian ini dilakukan melalui serangkaian tahapan sistematis untuk membangun, menguji, dan memanfaatkan program MATLAB dalam menghitung *Peirce Skill Score* (PSS) berbasis *Confusion matrix* 3×3.

2.2.1 Pembuatan Program *Peirce Skill Score*

Berikut adalah penjelasan tahapan pembuatan program *Peirce Skill Score* menggunakan matlab.

1. Membuat Fungsi “calculate_pss_with_epss” dan “calculate_metrics”

Langkah pertama dalam pembuatan program adalah membuat dua buah fungsi untuk menghitung metrik evaluasi. Fungsi pertama, calculate_pss_with_epss, dikembangkan untuk menghitung nilai *Peirce Skill Score* (PSS) sekaligus standard error-nya (ePSS). Perhitungan dilakukan berdasarkan distribusi probabilitas dari confusion matrix 3×3 yang dinormalisasi terhadap jumlah total observasi. Selain menghitung skor utama, fungsi ini juga menggunakan metode delta (Delta Method) untuk menurunkan turunan parsial terhadap setiap elemen matriks, sehingga dapat dihitung estimasi varians PSS dan
ılai ePSS. Nilai ePSS ini penting untuk menunjukkan tingkat
tabilitas dari skor PSS yang dihasilkan.

ılate_metrics, ditujukan untuk menghitung metrik evaluasi dari
ori setelah matriks 3×3 dikonversi menjadi tiga matriks 2×2. Metrik
i *Threat Score* (TS), *Bias* (B), *False Alarm Rate* (FAR), *Hit Rate* (H),
), PSS dan ePSS. Fungsi ini memastikan bahwa setiap kategori

dievaluasi secara independen sehingga kekuatan dan kelemahan model pada masing-masing jenis kejadian dapat teridentifikasi secara spesifik.

2. Inisialisasi Tabel Kontingensi 3x3

Setelah fungsi-fungsi dasar tersedia, langkah berikutnya adalah inisialisasi atau memasukkan data tabel kontingensi 3x3 yang mencerminkan hasil prakiraan terhadap observasi untuk tiga kategori curah hujan. Tabel ini diinput dalam bentuk matriks 3 baris dan 3 kolom, dengan setiap baris mewakili kategori hasil prakiraan dan setiap kolom mewakili kategori hasil observasi. Misalnya, baris pertama menunjukkan jumlah kejadian yang diprediksi sebagai "*No Rain*" dan kolom pertama menunjukkan kejadian yang diamati sebagai "*No Rain*". Tabel ini adalah dasar untuk semua perhitungan evaluasi selanjutnya.

3. Menghitung Nilai PSS dan ePSS

Setelah tabel kontingensi tersedia, fungsi calculate_pss_with_epss` dipanggil dengan parameter tabel tersebut. Nilai yang dihasilkan merupakan indikator akurasi sistem prakiraan secara keseluruhan, dengan memperhitungkan kemungkinan tebakan acak. PSS bernilai antara -1 hingga 1, dengan nilai 1 menunjukkan prakiraan sempurna, 0 menunjukkan prakiraan tidak lebih baik dari tebakan acak, dan nilai negatif menunjukkan prakiraan yang lebih buruk dari tebakan acak.

4. Menghitung Metrik Evaluasi per Kategori

Untuk mengevaluasi performa prakiraan pada masing-masing kategori secara individual, dilakukan pembentukan tabel kontingensi 2x2 untuk setiap kategori. Setiap tabel 2x2 dibentuk dengan mengambil nilai *True Positive* (TP), *False Positive* (FP), *False Negative* (FN), dan *True Negative* (TN) dari tabel 3x3. Nilai-nilai ini kemudian digunakan dalam fungsi `calculate_metrics` untuk memperoleh metrik evaluasi seperti TS, B, FAR, dan lainnya. Proses ini dilakukan dalam loop untuk ketiga kategori, sehingga menghasilkan evaluasi terpisah yang menggambarkan seberapa baik model memprediksi masing-masing fenomena.

5. Visualisasi Tabel dan Metrik

Langkah terakhir adalah melakukan visualisasi hasil evaluasi dalam bentuk teks yang diposisikan pada kanvas grafik (figure). Visualisasi mencakup: tampilan tabel kontingensi 3x3 secara utuh dengan nilai PSS, serta tabel 2x2 dan metrik evaluasi untuk masing-masing kategori. Semua teks ditampilkan menggunakan *font monospaced* agar penyusunan kolom tetap rapi dan sejajar. Visualisasi ini bertujuan untuk memberikan ringkasan yang jelas dan informatif mengenai kinerja sistem prakiraan, serta memudahkan pembaca dalam memahami hasil verifikasi klasifikasi.

ısil Program dengan Data Utama

metrik evaluasi dari program dibandingkan dengan nilai yang utama, yaitu pada Gambar 18. Penyamaan ini bertujuan untuk rogram yang dibuat mampu mereproduksi nilai-nilai evaluasi yang gan data acuan.

2.2.3 Penerapan Program pada Data Uji

Setelah validasi dilakukan, program kemudian diterapkan pada data uji pada studi kasus, dengan cara memasukkan nilai tabel *Confusion matrix* yang diperoleh dari studi tersebut. Tabel tersebut digunakan sebagai *input* ke dalam program untuk dilakukan perhitungan terhadap nilai PSS dan metrik evaluasi lainnya.

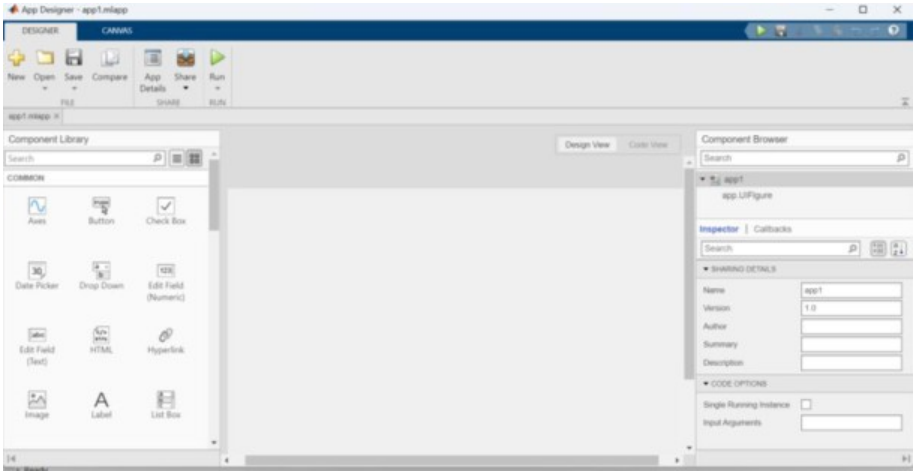
2.2.4 Analisis Hasil Evaluasi Data Uji

Tahap ini melakukan analisis terhadap hasil evaluasi dari data studi kasus. Nilai-nilai yang dihasilkan dibandingkan dengan interpretasi yang diberikan oleh penulis jurnal, dan dievaluasi untuk melihat seberapa baik sistem prakiraan yang digunakan dalam jurnal tersebut. Analisis ini juga memberikan insight mengenai kekuatan dan kelemahan model berdasarkan metrik objektif.

2.2.5 Implementasi GUI MATLAB

Sebagai bagian dari pengembangan sistem komputasi, program MATLAB yang telah dibuat diimplementasikan ke dalam bentuk antarmuka grafis (*Graphical User Interface*/ GUI). Implementasi GUI ini dilakukan dengan merancang tampilan interaktif yang memungkinkan pengguna memasukkan nilai-nilai tabel kontingensi 3x3 melalui *numeric input fields*, menghitung nilai PSS dan metrik evaluasi hanya dengan satu klik tombol, serta menampilkan hasil dalam bentuk teks yang informatif. Langkah-langkah implementasi GUI dilakukan sebagai berikut:

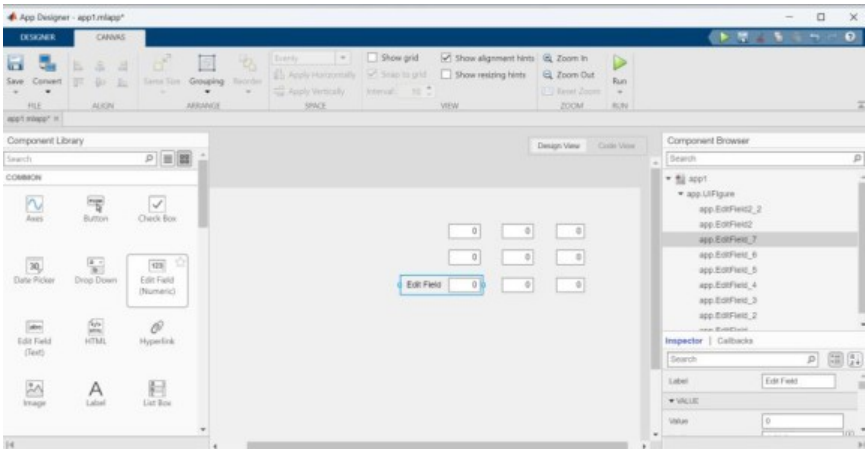
1. Membuat layout GUI menggunakan *MATLAB App Designer*.



Gambar 22. Tampilan awal membuat *layout GUI* menggunakan *MATLAB App Designer*

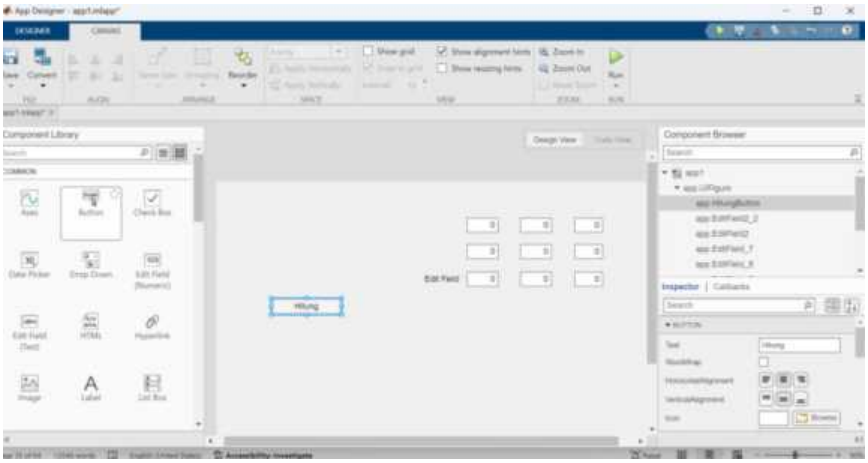
ıponen utama dalam GUI, yaitu:

3×3 menggunakan *Numeric Edit Fields*, yang diatur dalam formasi lom untuk merepresentasikan kategori prediksi dan observasi pada ıtrix.



Gambar 23. Memasukkan *Numeric Edit Field* pada *layout*

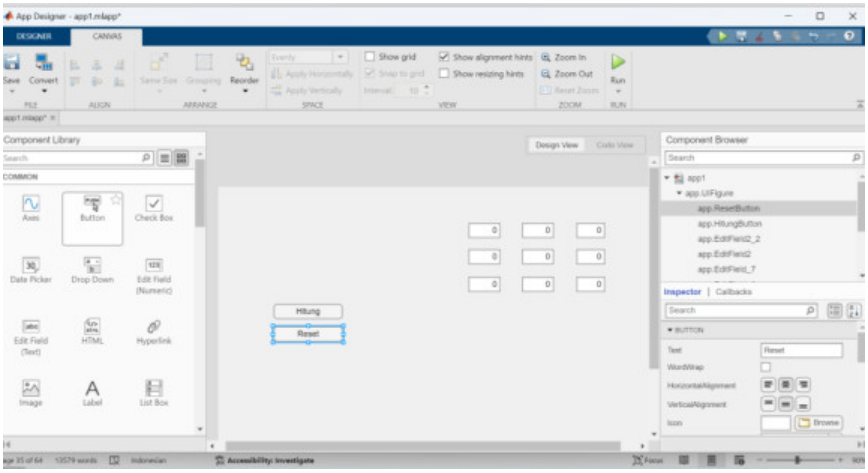
- Tombol eksekusi perhitungan, yang berfungsi untuk menjalankan proses penghitungan nilai PSS dan metrik evaluasi ketika ditekan.



Gambar 24. Memasukkan tombol hitung pada *layout*

Pada callbacks tombol Hitung, dimasukkan script program Peirce Skill Score (PSS) yang telah dibuat sebelumnya, serta menghubungkannya dengan matriks 3×3 yang telah dibuat sebagai input dan text area sebagai output tempat hasil perhitungan ditampilkan. Proses ini memastikan bahwa setiap kali tombol ditekan, sistem akan secara otomatis membaca data input, menghitung nilai PSS beserta metrik evaluasi lainnya, dan menyajikan hasilnya dalam format tabel evaluasi yang terstruktur di tampilan GUI. Hal ini mempermudah pengguna ukan verifikasi prediksi secara cepat dan akurat.

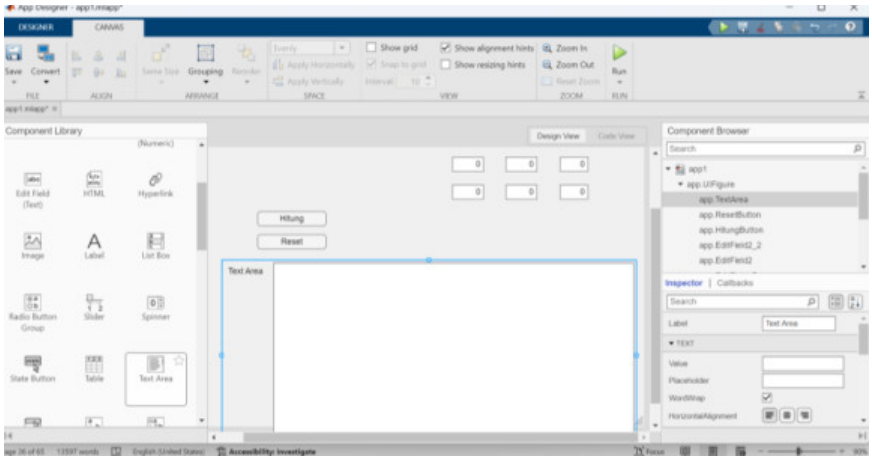
, yang digunakan untuk menghapus atau mengosongkan seluruh *utput* agar pengguna dapat memulai perhitungan baru dengan



Gambar 25. Memasukkan tombol reset pada *layout*

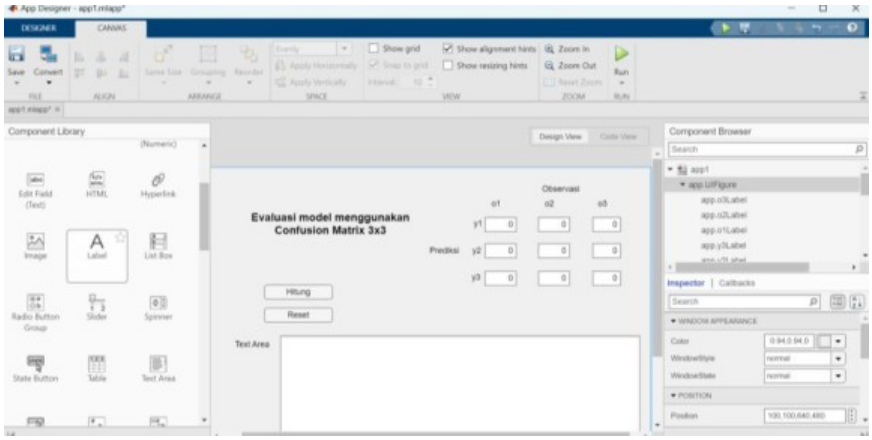
Pada *callbacks* tombol Reset, fungsi dihubungkan dengan matriks 3×3 dan text area, sehingga ketika tombol ditekan, seluruh nilai input pada matriks akan dikosongkan dan area output akan dibersihkan. Dengan demikian, pengguna dapat melakukan perhitungan ulang dari awal tanpa harus menutup atau memuat ulang aplikasi. Hal ini meningkatkan efisiensi dan kenyamanan dalam proses verifikasi.

- Display khusus untuk nilai PSS dan metrik pendukung, ditampilkan dalam bentuk *text area* yang otomatis diperbarui setelah perhitungan dilakukan.



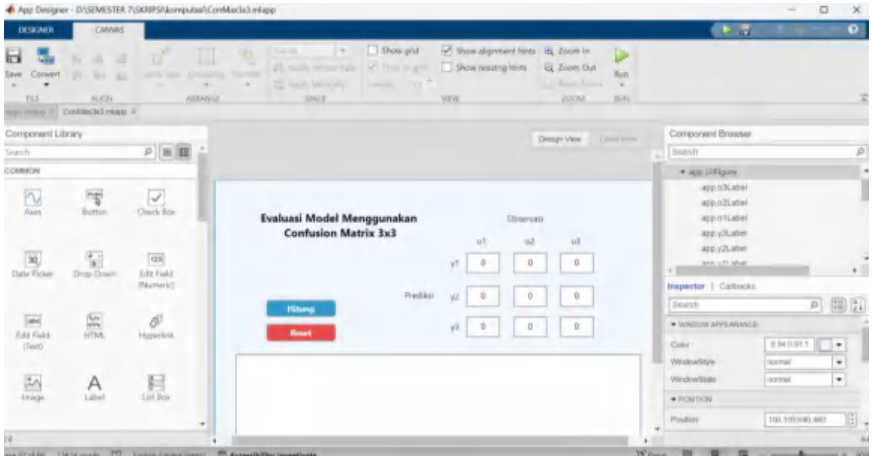
Gambar 26. Memasukkan *text area* pada layout

ın judul dan keterangan lainnya dengan *Label.*
ɔerjelas tampilan, ditambahkan judul dan keterangan lainnya
n komponen Label, yang berfungsi sebagai penanda masing-
ın input, output, dan fungsi tombol pada antarmuka GUI.



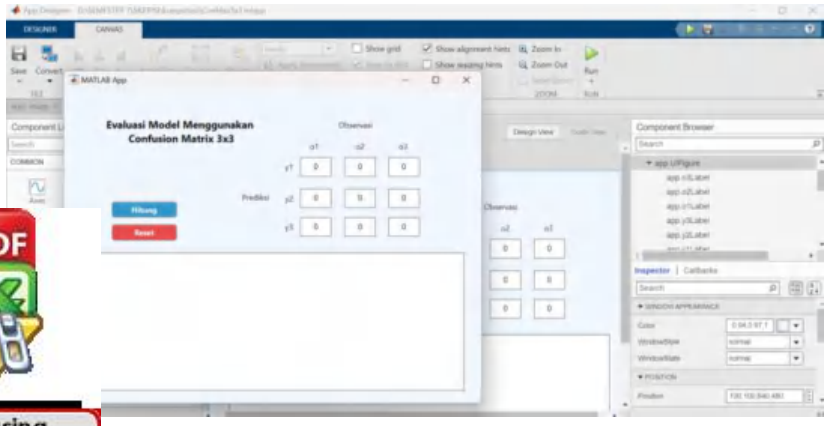
Gambar 27. Menambahkan judul dan keterangan pada *layout*

- Menambahkan warna pada *layout* dan juga komponen agar tampilan lebih menarik.



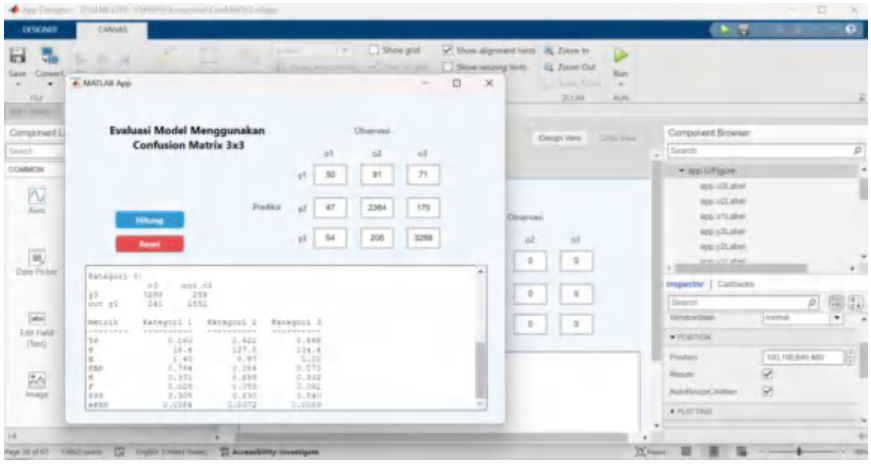
Gambar 28. Menambahkan warna pada layout dan komponen

- Setelah selesai mendesain aplikasi nya dan telah memasukan *script* perhitungan PSS, langkah selanjutnya adalah *run* aplikasi.



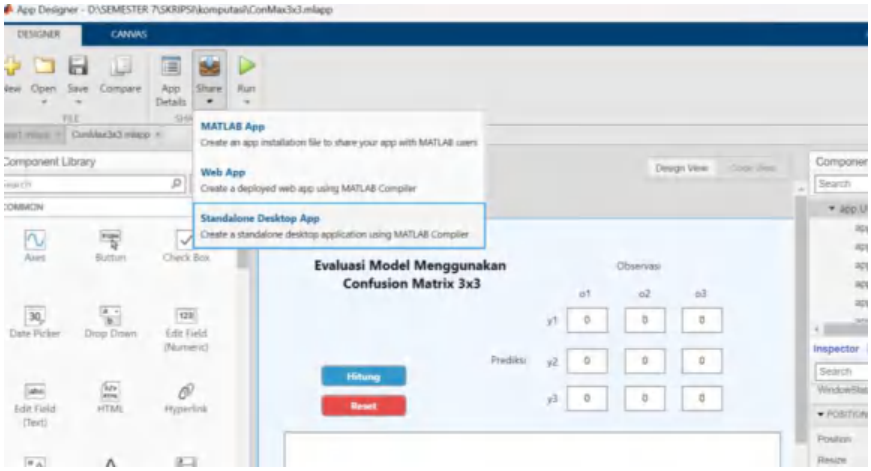
Gambar 29. Tampilan setelah run aplikasi

- Setelah itu melakukan pengecekan dengan memasukan nilai pada data yang digunakan. Serta memastikan bahwa setiap komponen dapat berfungsi dengan baik.



Gambar 30. Pengecekan hasil perhitungan

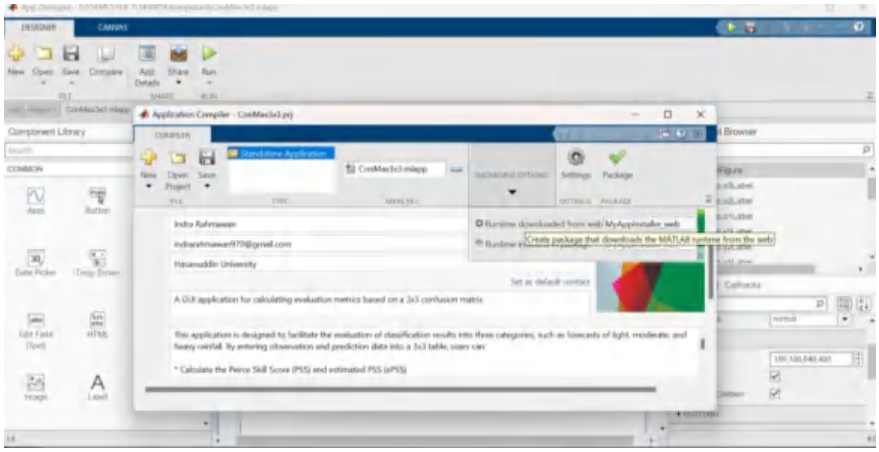
- Setelah melakukan pengecekan hasil perhitungan dan juga komponen, langkah selanjutnya adalah mengekstrak aplikasi menjadi file exe menggunakan *MATLAB Compiler* dengan memilih opsi *Standole Dekstop App* sehingga aplikasi dapat dibagikan.



Gambar 31. Mengekstrak aplikasi menggunakam *MATLAB Compiler*

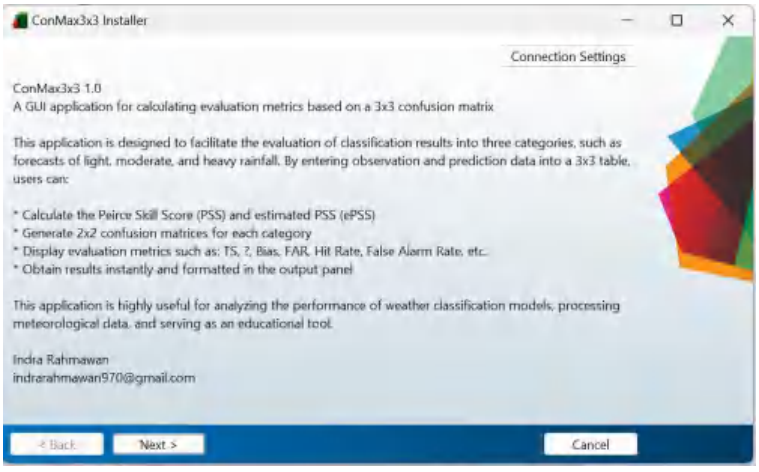
- Selanjutnya, masukkan data informasi yang dibutuhkan seperti nama aplikasi, nbang, nama institusi, serta ringkasan dan deskripsi aplikasi pada rsedia di jendela *App Packaging*. Informasi ini akan ditampilkan adata saat aplikasi dijalankan. Pada bagian *Packaging Options*, *ntime downloaded from web*" agar pengguna dapat secara otomatis MATLAB Runtime jika belum terinstal di komputernya. Setelah masi lengkap dan opsi dikonfigurasi, klik tombol *Package* untuk ses *bundling* dan ekstraksi. Proses ini akan menghasilkan file

instalasi yang siap digunakan sebagai aplikasi mandiri (*standalone*), yang dapat dijalankan tanpa perlu membuka MATLAB.

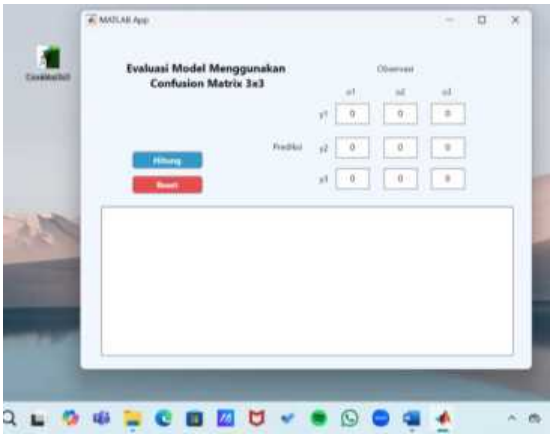


Gambar 32. Proses *package* aplikasi

- Setelah proses ekstrasi aplikasi, maka aplikasi siap di *install* dan siap digunakan.



Gambar 33. Proses instalasi aplikasi



Gambar 34. Tampilan aplikasi pada dekstop

2.3 Bagan Alir