

Laporan Ujian Akhir Semester
“Analisis Data Spasial untuk Deteksi Tingkat Keparahannya Gempa Bumi di Indonesia Menggunakan Machine Learning”



Dosen Pengampu :

Dr. Elly Matul Imah, M.Kom
Ulfa Siti Nuraini, S.Stat., M.Stat

Disusun Oleh (Kelompok 5) :

Hendra Cahyono	22031554029
Joevita Salsabila Fitrianova	22031554031
Titania Aurera Nur Azima	22031554016

S1 SAINS DATA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI SURABAYA

2024

Pendahuluan

Latar Belakang

Indonesia sebagai salah satu negara yang terletak di wilayah Cincin Api Pasifik, memiliki risiko tinggi terhadap kejadian gempa bumi. Keberadaan beberapa lempeng tektonik yang saling bertemu di bawah wilayah Indonesia menyebabkan negara ini sering mengalami aktivitas seismik yang signifikan. Kejadian gempa bumi ini seringkali menimbulkan kerugian materiil dan korban jiwa yang tidak sedikit. Oleh karena itu, mitigasi bencana gempa bumi menjadi hal yang sangat penting untuk diperhatikan oleh pemerintah dan pihak terkait.

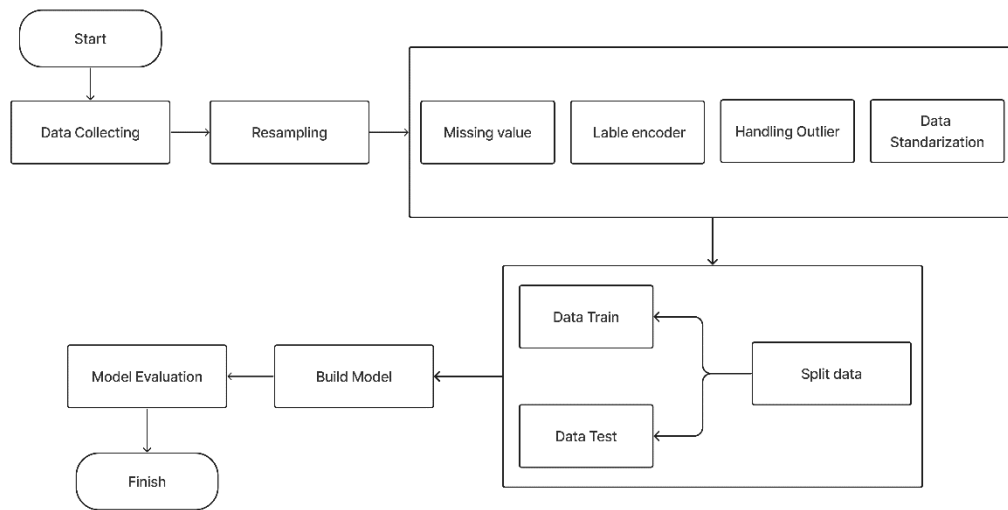
Salah satu langkah mitigasi yang dapat dilakukan adalah dengan mengembangkan sistem prediksi yang akurat untuk memprediksi tingkat keparahan gempa bumi. Dalam konteks ini, data spasial memiliki peran yang sangat penting. Data spasial mencakup informasi mengenai lokasi episenter gempa, karakteristik tanah, struktur geologi, serta pola distribusi populasi yang bisa terpengaruh oleh gempa tersebut. Analisis data spasial ini dapat memberikan pemahaman yang lebih mendalam mengenai potensi dampak dari gempa bumi, sehingga memungkinkan pihak berwenang untuk melakukan perencanaan yang lebih baik dalam menghadapi bencana.

Dalam beberapa tahun terakhir, metode machine learning telah banyak digunakan untuk analisis data spasial dalam konteks mitigasi bencana. Dua metode yang efektif dalam hal ini adalah algoritma Random Forest dan Decision Tree. Algoritma Random Forest adalah metode ensemble yang menggabungkan beberapa pohon keputusan untuk meningkatkan akurasi prediksi. Setiap pohon keputusan dalam Random Forest dilatih dengan subset data yang berbeda, sehingga dapat mengurangi overfitting dan memberikan hasil prediksi yang lebih stabil dan akurat. Di sisi lain, algoritma Decision Tree membentuk model prediktif berdasarkan aturan-aturan yang dihasilkan dari data. Metode ini sangat intuitif dan mudah diinterpretasikan, karena setiap keputusan dalam model dapat ditelusuri kembali ke aturan-aturan yang mendasarinya.

Proyek "Analisis Data Spasial untuk Prediksi Tingkat Keparahannya Gempa Bumi di Indonesia Menggunakan Algoritma Random Forest dan Decision Tree" bertujuan untuk mengembangkan model prediktif yang dapat membantu pemerintah dan badan terkait dalam meningkatkan kesiapsiagaan menghadapi gempa bumi. Dengan model prediktif yang akurat, diharapkan dapat dilakukan perencanaan infrastruktur yang lebih tahan gempa dan penyusunan strategi mitigasi bencana yang lebih efektif. Selain itu, model ini juga diharapkan dapat memberikan informasi yang berguna bagi masyarakat untuk meningkatkan kesadaran dan kesiapsiagaan terhadap potensi gempa bumi.

Pengembangan model ini melibatkan beberapa tahapan, mulai dari pengumpulan dan preprocessing data spasial, pemilihan fitur yang relevan, pelatihan dan evaluasi model menggunakan algoritma Random Forest dan Decision Tree, hingga interpretasi hasil prediksi untuk memberikan rekomendasi yang aplikatif. Dengan pendekatan ini, diharapkan proyek ini dapat memberikan kontribusi nyata dalam upaya mitigasi bencana gempa bumi di Indonesia, serta mendukung pembangunan yang lebih aman dan berkelanjutan di masa mendatang.

Hasil dan Pembahasan



Gambar 1. Flow Chart Model building

Flow chart tersebut merupakan alur pembuatan Analisis Data Spasial untuk Deteksi Tingkat Keparahan Gempa Bumi di Indonesia Menggunakan Machine Learning

Dengan Resampling

1. Data Collecting

Dataset yang digunakan untuk pelatihan model (train model) yakni dataset gempa bumi yang bersumber dari kaggle <https://www.kaggle.com/datasets/warcoder/earthquake-dataset> dengan 19 kolom dan 782 data atau baris. Dataset kedua yakni dataset yang digunakan untuk pengujian (data test) yang berasal dari situs resmi Badan Meteorologi, Klimatologi dan Geofisika (BMKG) <https://repogempa.bmkg.go.id/eventcatalog> dengan jumlah kolom sebanyak 12 kolom dan 666 data atau baris. Dataset yang digunakan bisa dilihat sebagai berikut :

	title	magnitude	date_time	cdi	mml	alert	tsunami	sig	net	nst	dmin	gap	magType	depth	latitude	longitude	location	continent	country
0	M 7.0 - 18 km SW of Malango, Solomon Islands	7.0	22-11-2022 02:03	8	7	green	1	768	us	117	0.509	17.0	mwv	14.000	-9.7963	159.596	Malango, Solomon Islands	Oceania	Solomon Islands
1	M 6.9 - 204 km SW of Bengkulu, Indonesia	6.9	18-11-2022 13:37	4	4	green	0	735	us	99	2.229	34.0	mwv	25.000	-4.9559	100.738	Bengkulu, Indonesia	NaN	NaN
2	M 7.0 -	7.0	12-11-2022 07:09	3	3	green	1	755	us	147	3.125	18.0	mwv	579.000	-20.0508	-178.346	NaN	Oceania	Fiji
3	M 7.3 - 205 km ESE of Neiafu, Tonga	7.3	11-11-2022 10:48	5	5	green	1	833	us	149	1.865	21.0	mwv	37.000	-19.2918	-172.129	Neiafu, Tonga	NaN	NaN
4	M 6.6 -	6.6	09-11-2022 10:14	0	2	green	1	670	us	131	4.998	27.0	mwv	624.464	-25.5948	178.278	NaN	NaN	NaN
...
777	M 7.7 - 28 km SSW of Puerto El Triunfo, El Sal...	7.7	13-01-2001 17:33	0	8	NaN	0	912	us	427	0.000	0.0	mwv	60.000	13.0490	-88.660	Puerto El Triunfo, El Salvador	NaN	NaN
778	M 6.9 - 47 km S of Old Harbor, Alaska	6.9	10-01-2001 16:02	5	7	NaN	0	745	ak	0	0.000	0.0	mwv	36.400	56.7744	-153.281	Old Harbor, Alaska	North America	NaN
779	M 7.1 - 16 km NE of Port-Olry, Vanuatu	7.1	09-01-2001 16:49	0	7	NaN	0	776	us	372	0.000	0.0	mwv	103.000	-14.9280	167.170	Port-Olry, Vanuatu	NaN	Vanuatu
780	M 6.8 - Mindanao, Philippines	6.8	01-01-2001 08:54	0	5	NaN	0	711	us	64	0.000	0.0	mwv	33.000	6.6310	126.899	Mindanao, Philippines	NaN	NaN
781	M 7.5 - 21 km SE of Lukatan, Philippines	7.5	01-01-2001 06:57	0	7	NaN	0	865	us	324	0.000	0.0	mwv	33.000	6.8980	126.579	Lukatan, Philippines	NaN	Philippines

Gambar 2. Dataset 1 (from kaggle)

No	Event ID	Date time	Latitude	Longitude	Magnitude	Mag Type	Depth (km)	Phase Count	Azimuth Gap	Location	Agency
0	1 bmg2024inoy	2024-05-01T06:35:52.433043Z	-1.710596	120.091766	2.837047	M	10	23	76.859665	Sulawesi, Indonesia	BMKG
1	2 bmg2024injv	2024-05-01T04:00:52.007557Z	-5.944668	130.647629	4.846796	MLv	133	48	78.901794	Banda Sea	BMKG
2	3 bmg2024injs	2024-05-01T03:57:30.664236Z	-6.438559	107.307701	2.836673	M	10	25	80.936186	Java, Indonesia	BMKG
3	4 bmg2024inia	2024-05-01T03:06:11.494109Z	-7.134665	107.558365	4.171044	M	10	69	83.403366	Java, Indonesia	BMKG
4	5 bmg2024ingp	2024-05-01T02:23:37.672776Z	-7.266338	129.565552	4.551028	M	166	68	55.587601	Banda Sea	BMKG
...
661	666 bmg2024gldq	2024-04-01T11:32:21.820523Z	-8.737351	109.740349	3.031500	M	12	22	237.116982	Java, Indonesia	BMKG
662	667 bmg2024gldb	2024-04-01T11:14:53.526143Z	4.079156	128.337433	4.160199	M	14	19	255.591034	North of Halmahera, Indonesia	BMKG
663	668 bmg2024glcw	2024-04-01T11:09:01.756269Z	-0.221772	123.710251	3.289224	M	54	27	87.837761	Minahassa Peninsula, Sulawesi	BMKG
664	669 bmg2024glbr	2024-04-01T10:32:54.397Z	-5.309146	102.562805	4.191902	M	13	29	156.701263	Southern Sumatra, Indonesia	BMKG
665	670 bmg2024glbl	2024-04-01T10:26:27.201897Z	-5.751341	112.520355	4.559490	M	10	64	87.903351	Java Sea	BMKG

Gambar 3. Dataset 2 (from BMKG)

2. Preprocessing

Preprocessing adalah tahap awal dalam pemrosesan data yang bertujuan untuk mempersiapkan data mentah sehingga siap digunakan dalam analisis atau model machine learning. langkah-langkah preprocessing (termasuk membaca dataset, memeriksa kualitas data, memilih kolom penting, menghapus nilai hilang, label encoding, deteksi dan penanganan outliers, serta normalisasi data) dilakukan terlebih dahulu. Setelah data telah dipreproses dan dibersihkan. Proses yang dilakukan yakni sebagai berikut :

- Memeriksa missing value dan data duplikat

1	<code>df.isna().sum()</code>
1	<code>## Memeriksa dataset</code>
2	<code>print(f'Number of records (rows) in the dataset are: {df.shape[0]}')</code>
3	<code>print(f'Number of features (columns) in the dataset are: {df.shape[1]}')</code>
4	<code>print(f'Number of duplicate entries in the dataset are: {df.duplicated().sum()}')</code>
5	<code>print(f'Number missing values in the dataset are: {sum(df.isna().sum())}')</code>
Number of records (rows) in the dataset are: 782 Number of features (columns) in the dataset are: 19 Number of duplicate entries in the dataset are: 0 Number missing values in the dataset are: 1246	

Gambar 4. Missing value

Memeriksa jumlah nilai yang hilang (missing values) dalam setiap kolom pada DataFrame, jumlah baris dan kolom, jumlah entri duplikat, dan jumlah nilai yang hilang dalam dataset.

- Memilih kolom yang relevan dan menghapus missing value

Dari proses sebelumnya yang telah dilakukan yakni memeriksa missing value, terdapat banyak NaN yang ditemukan pada kolom label (alert) yang akan digunakan untuk prediksi label. Oleh karena itu, untuk mengatasi hal tersebut dilakukan penghapusan NaN yang terdapat pada kolom label (alert) agar tidak ada data yang tidak mempunyai label. Langkah selanjutnya yakni memilih kolom-kolom yang relevan dan sesuai dengan data test yang akan digunakan nantinya yaitu data test bmkg. Kolom-kolom yang dipilih yakni 8 kolom dengan 7 kolom sebagai fitur dan 1 kolom sebagai kolom label.

```
1 data = df[['latitude', 'longitude', 'magnitude', 'magType', 'depth', 'nst', 'gap', 'alert']]
2 data
```

	latitude	longitude	magnitude	magType	depth	nst	gap	alert
0	-9.7963	159.596	7.0	mww	14.000	117	17.0	green
1	-4.9559	100.738	6.9	mww	25.000	99	34.0	green
2	-20.0508	-178.346	7.0	mww	579.000	147	18.0	green
3	-19.2918	-172.129	7.3	mww	37.000	149	21.0	green
4	-25.5948	178.278	6.6	mww	624.464	131	27.0	green
...
416	-28.0940	-70.653	6.8	mww	45.000	596	19.3	green
423	13.9880	-91.895	7.4	mww	24.000	751	25.5	yellow
440	0.8020	92.463	8.2	mwc	25.100	341	14.9	green
441	2.3270	93.063	8.6	mw	20.000	499	16.6	yellow
507	32.2862	-115.295	7.2	mw	9.987	10	239.0	red

415 rows x 8 columns

Gambar 4. Pemilihan kolom yang relevan

- Label Encoder

Mengubah data kategori pada kolom magType dan kolom alert menjadi data numerik menggunakan LabelEncoder.

	latitude	longitude	magnitude	magType	depth	nst	gap	alert
0	-9.7963	159.5960	7.0	4	14.0	117	17.0	0
1	-4.9559	100.7380	6.9	4	25.0	99	34.0	0
2	-20.0508	-178.3460	7.0	4	579.0	147	18.0	0
3	-19.2918	-172.1290	7.3	4	37.0	149	21.0	0
7	7.6712	-82.3396	6.7	4	20.0	145	37.0	0

Gambar 5. Label Encoder

- Menangani Outliers

```

14
15 # Memilih hanya kolom-kolom numerik
16 numeric_columns = dt.select_dtypes(include=[np.number])
17
18 # Deteksi outliers menggunakan Z-score pada kolom numerik
19 z_scores = stats.zscore(numeric_columns)
20 abs_z_scores = np.abs(z_scores)
21 filtered_entries = (abs_z_scores < 3).all(axis=1)
22 print("Outliers berdasarkan Z-Score:")
23 dt[filtered_entries]

```

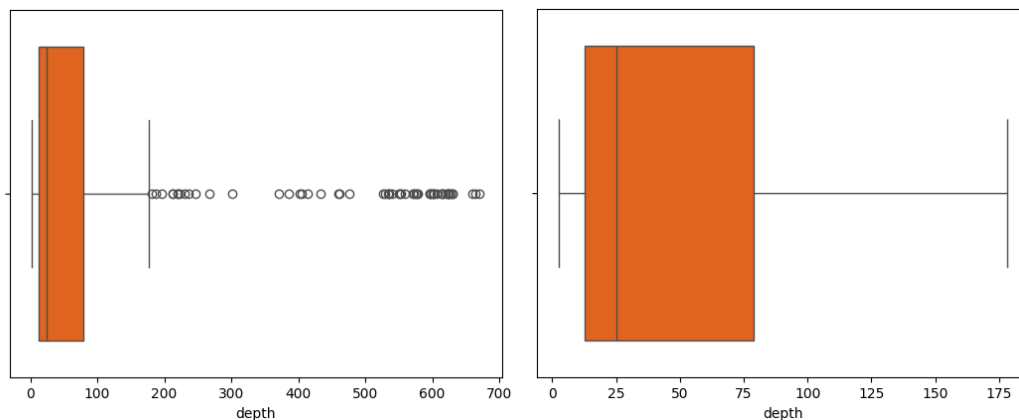
Mengidentifikasi outliers dengan Z-score dan menghapus baris yang mengandung outliers.

```

1 def ganti_outlier_batas_ekstrim(st, kolom):
2     q1 = st[kolom].quantile(0.25)
3     q3 = st[kolom].quantile(0.75)
4     rentang_interkuartil = q3 - q1
5
6     batas_atas = q3 + (rentang_interkuartil * 1.5)
7     batas_bawah = q1 - (rentang_interkuartil * 1.5)
8
9     st[kolom].mask(st[kolom] > batas_atas, batas_atas, inplace=True)
10    st[kolom].mask(st[kolom] < batas_bawah, batas_bawah, inplace=True)

```

Pada code tersebut mengganti outlier dengan batas ekstrimnya, sebelumnya juga telah dihitung berapa persen outlier pada masing masing fitur. latitude : 0.00 % longitude : 0.00 % magnitude : 4.34 % magType : 3.61 % depth : 13.98 % nst : 14.70 % gap : 6.51 % alert : 21.69 %.



Gambar 6. Dataset sebelum dan sesudah penanganan outliers

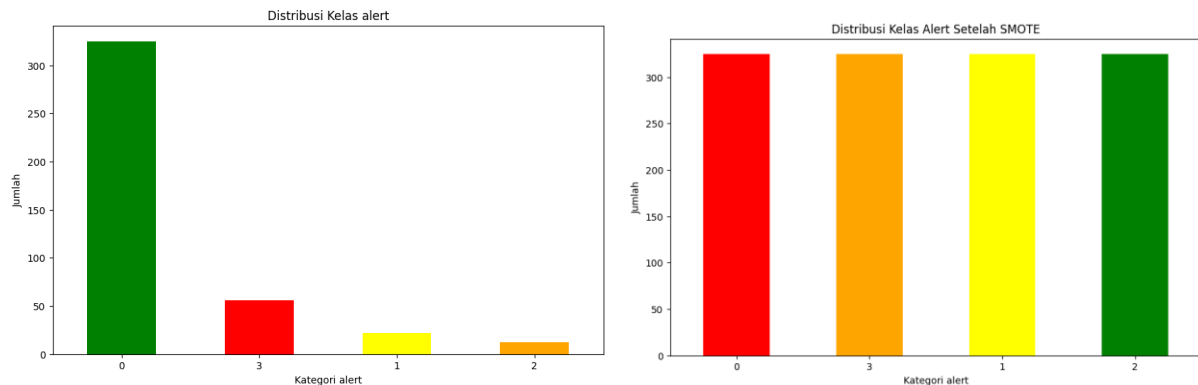
- Standarisasi Data

...	latitude	longitude	magnitude	magType	depth	nst	gap
0	-0.374921	0.973224	0.263032	0.181898	-0.693701	0.510171	-0.550910
1	-0.207412	0.505608	0.004363	0.181898	-0.510907	0.379569	0.734449
2	-0.729790	-1.711663	0.263032	0.181898	2.036698	0.727842	-0.475300
3	-0.703524	-1.662270	1.039040	0.181898	-0.311496	0.742353	-0.248472
4	-0.921647	1.121649	-0.771645	0.181898	2.036698	0.611751	0.205184
...
410	-1.008135	-0.856062	-0.254306	0.181898	-0.178555	3.985646	-0.377008
411	0.448165	-1.024826	1.297709	0.181898	-0.527525	5.110278	0.091770
412	-0.008153	0.439865	2.461721	-1.858307	-0.509245	2.135446	-0.709689
413	0.044621	0.444632	2.461721	-5.938718	-0.593995	3.281845	-0.581154
414	1.081397	-1.210734	0.780371	-5.938718	-0.760387	-0.266188	2.322246

Gambar 7. Standarisasi Data

3. Resampling

Resampling adalah teknik yang digunakan dalam analisis data dan machine learning untuk mengubah ukuran atau distribusi data dalam dataset. Tujuan utama resampling adalah untuk mengatasi masalah ketidakseimbangan kelas, meningkatkan generalisasi model, atau mengevaluasi performa model secara lebih andal. Ada dua jenis resampling yang umum digunakan: Disini kami menggunakan SMOTE (Synthetic Minority Over-sampling Technique) untuk Menciptakan sampel-sampel sintetis baru dalam kelas minoritas dengan menginterpolasi antara sampel-sampel yang ada. SMOTE digunakan untuk mengatasi ketidakseimbangan kelas pada kolom alert.



Gambar 8. Distribusi kelas pada dataset

4. Split Data

```
1 from sklearn.model_selection import train_test_split
2
3 # Membagi data menjadi training dan testing set
4 X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=42)
```

Membagi dataset menjadi training dan testing set untuk evaluasi model.

5. Model Machine Learning (Random Forest, Logistic Regression, Decision Tree, SVM, KNN)

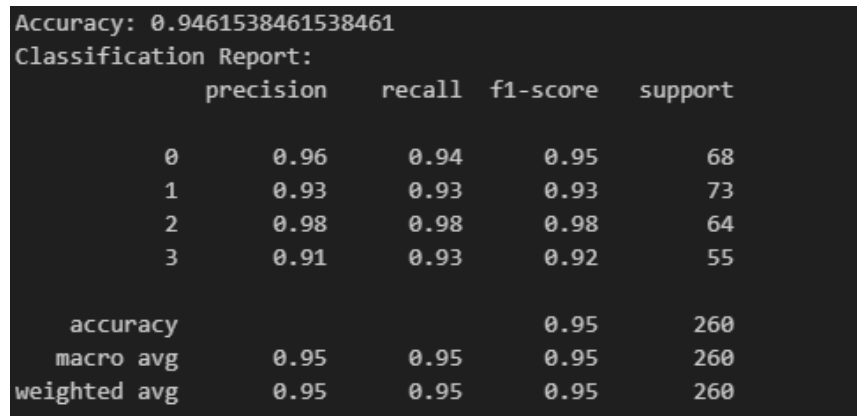
5.1 Random Forest

```
1 from sklearn.ensemble import RandomForestClassifier
2 from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
3
4 # Inisialisasi model
5 model1 = RandomForestClassifier(random_state=42)
6
7 # Latih model dengan data training
8 model1.fit(X_train, y_train)
9
10 # Prediksi dengan data testing
11 y_pred = model1.predict(X_test)
12 # Evaluasi kinerja model
13 print("Accuracy:", accuracy_score(y_test, y_pred))
14 print("Classification Report:")
15 print(classification_report(y_test, y_pred))
```

Random Forest adalah algoritma ensemble learning yang terdiri dari sejumlah besar decision trees individual yang bekerja bersama untuk menghasilkan prediksi yang lebih akurat dan kuat. Metode ini digunakan untuk tugas klasifikasi dan regresi. Setiap tree dalam hutan dibangun dari subset acak dari data pelatihan dengan menggunakan pengambilan sampel bootstrap, dan setiap split pada node dibuat dari subset acak dari fitur yang dipilih. Hasil akhir dari Random Forest diperoleh dengan menggabungkan prediksi dari semua trees, biasanya melalui voting

mayoritas untuk klasifikasi atau rata-rata untuk regresi. Pendekatan ini membantu mengurangi overfitting dan meningkatkan generalisasi model karena variasi antar trees membantu menyeimbangkan kelemahan masing-masing tree individual. Random Forest juga memberikan estimasi pentingnya fitur, sehingga berguna dalam fitur seleksi.

Hasil:

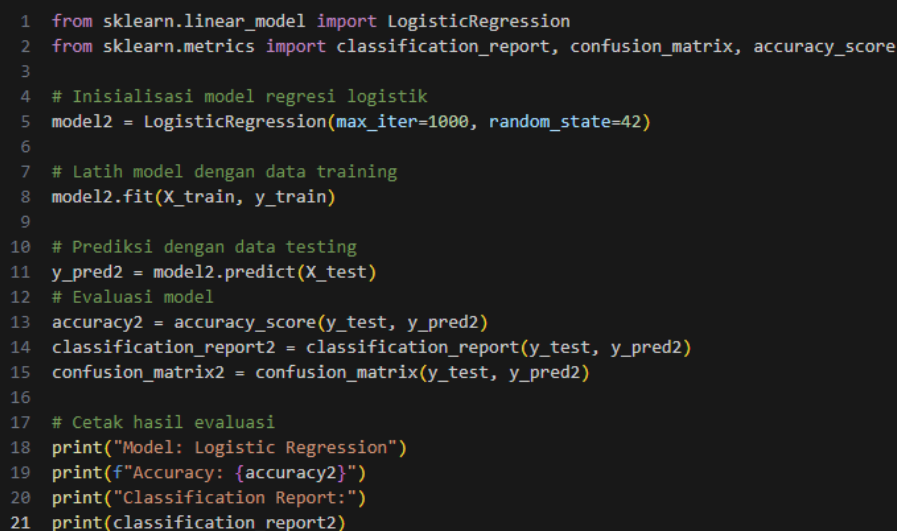


```
Accuracy: 0.9461538461538461
Classification Report:
```

	precision	recall	f1-score	support
0	0.96	0.94	0.95	68
1	0.93	0.93	0.93	73
2	0.98	0.98	0.98	64
3	0.91	0.93	0.92	55
accuracy			0.95	260
macro avg	0.95	0.95	0.95	260
weighted avg	0.95	0.95	0.95	260

Gambar 9. Hasil akurasi RF setelah resampling

5.2 Logistic Regression



```
1 from sklearn.linear_model import LogisticRegression
2 from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
3
4 # Inisialisasi model regresi logistik
5 model2 = LogisticRegression(max_iter=1000, random_state=42)
6
7 # Latih model dengan data training
8 model2.fit(X_train, y_train)
9
10 # Prediksi dengan data testing
11 y_pred2 = model2.predict(X_test)
12 # Evaluasi model
13 accuracy2 = accuracy_score(y_test, y_pred2)
14 classification_report2 = classification_report(y_test, y_pred2)
15 confusion_matrix2 = confusion_matrix(y_test, y_pred2)
16
17 # Cetak hasil evaluasi
18 print("Model: Logistic Regression")
19 print(f"Accuracy: {accuracy2}")
20 print("Classification Report:")
21 print(classification_report2)
```

Logistic regression adalah metode statistik yang digunakan untuk memodelkan probabilitas kejadian suatu peristiwa biner (dua kemungkinan) berdasarkan satu atau lebih variabel independen. Dalam konteks machine learning, logistic regression adalah algoritma klasifikasi yang memprediksi nilai kategori (biasanya 0 atau 1) dengan menggunakan fungsi logistik untuk mengubah output linear dari variabel-variabel input menjadi probabilitas antara 0 dan 1.

Hasil:

Model: Logistic Regression				
Accuracy: 0.6384615384615384				
Classification Report:				
	precision	recall	f1-score	support
0	0.71	0.72	0.72	68
1	0.64	0.56	0.60	73
2	0.70	0.81	0.75	64
3	0.45	0.44	0.44	55
accuracy			0.64	260
macro avg	0.63	0.63	0.63	260
weighted avg	0.63	0.64	0.63	260

Gambar 10. Hasil akurasi logreg setelah resampling

5.3 Decision Tree

Algoritma ini secara berulang mempartisi dataset ke dalam subset yang lebih homogen berdasarkan nilai fitur yang memaksimalkan pemisahan, seperti dengan mengurangi entropi atau meningkatkan informasi, dengan tugas klasifikasi dan regresi yang memisahkan data berdasarkan fitur-fitur dalam bentuk struktur pohon.

```

1 from sklearn.tree import DecisionTreeClassifier
2 from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
3
4 # Inisialisasi model Decision Tree
5 model3 = DecisionTreeClassifier(random_state=42)
6
7 # Latih model dengan data training
8 model3.fit(X_train, y_train)
9
10 # Prediksi dengan data testing
11 y_pred3 = model3.predict(X_test)

```

Hasil:

Model: Decision Tree				
Accuracy: 0.8346153846153846				
Classification Report:				
	precision	recall	f1-score	support
0	0.87	0.79	0.83	68
1	0.80	0.84	0.82	73
2	0.93	0.89	0.91	64
3	0.74	0.82	0.78	55
accuracy			0.83	260
macro avg	0.84	0.83	0.83	260
weighted avg	0.84	0.83	0.84	260

Gambar 11. Hasil akurasi decision tree setelah resampling

5.4 SVM

SVM bekerja dengan mencari hyperplane terbaik yang memisahkan data ke dalam kelas yang berbeda dengan margin terbesar. Dalam kasus data yang tidak dapat dipisahkan secara linear, SVM menggunakan kernel trick untuk memetakan data ke dimensi yang lebih tinggi di mana pemisahan menjadi lebih mudah. SVM sangat efektif dalam ruang dimensi tinggi dan dapat menangani non-linearitas dengan baik. Namun, SVM bisa lambat dalam kasus dataset yang sangat besar.

```

1 from sklearn.svm import SVC
2 from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
3
4 # Inisialisasi model SVM
5 model4 = SVC(random_state=42)
6
7 # Latih model dengan data training
8 model4.fit(X_train, y_train)
9
10 # Prediksi dengan data testing
11 y_pred4 = model4.predict(X_test)

```

Hasil:

Model: Support Vector Machine					
Accuracy: 0.8576923076923076					
Classification Report:					
	precision	recall	f1-score	support	
0	0.84	0.85	0.85	68	
1	0.85	0.82	0.83	73	
2	0.93	0.98	0.95	64	
3	0.81	0.76	0.79	55	
accuracy			0.86	260	
macro avg	0.85	0.86	0.85	260	
weighted avg	0.86	0.86	0.86	260	

Gambar 12. Hasil akurasi SVM setelah resampling

5.5 K-Nearest Neighbor (KNN)

adalah algoritma klasifikasi yang bekerja dengan membandingkan titik data baru dengan k tetangga terdekat dalam dataset pelatihan. Klasifikasi ditentukan berdasarkan mayoritas label dari tetangga-tetangga tersebut. KNN mudah diimplementasikan dan intuitif, namun bisa menjadi lambat dengan dataset yang besar karena memerlukan perhitungan jarak untuk setiap prediksi. Algoritma ini juga sensitif terhadap skala data dan keberadaan fitur yang tidak relevan.

```

1 from sklearn.neighbors import KNeighborsClassifier
2 from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
3
4 # Inisialisasi model K-Nearest Neighbors
5 model5 = KNeighborsClassifier()
6
7 # Latih model dengan data training
8 model5.fit(X_train, y_train)
9
10 # Prediksi dengan data testing
11 y_pred5 = model5.predict(X_test)

```

Hasil:

```

Model: K-Nearest Neighbors
Accuracy: 0.8884615384615384
Classification Report:

```

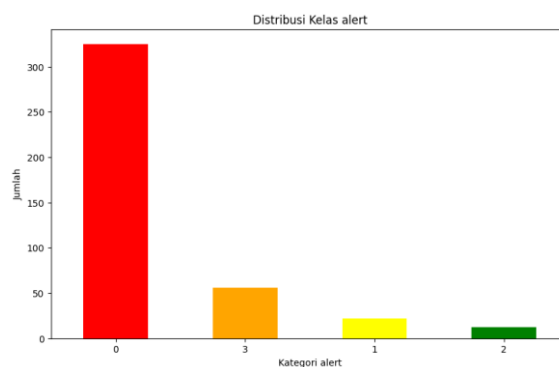
	precision	recall	f1-score	support
0	0.98	0.76	0.86	68
1	0.85	0.92	0.88	73
2	0.93	1.00	0.96	64
3	0.81	0.87	0.84	55
accuracy			0.89	260
macro avg	0.89	0.89	0.89	260
weighted avg	0.90	0.89	0.89	260

Gambar 13. Hasil akurasi KNN setelah resampling

Karena model paling baik menggunakan decision Tree, maka model disimpan untuk pengujian. Pemredelan juga dilakukan percobaan tanpa Resampling. Untuk langkah Preprocessing sama dengan yang menggunakan resampling, maka akan dijelaskan langsung pada bagian yang berbeda. Berikut merupakan langkah-langkah model tanpa resampling:

Tanpa Resampling

1. Distribusi Kelas



Gambar 14. Distribusi kelas sebelum resampling

Berdasarkan gambar tersebut terlihat sekali jika persebaran atau distribusi kelas pada dataset sangat tidak seimbang atau imbalance. Oleh karena itu, pada proses sebelumnya kami melakukan resampling pada dataset untuk menghindari adanya overfitting dan juga untuk meningkatkan performa model.

6. Model Machine Learning (Random Forest, Logistic Regression, Decision Tree, SVM, KNN)

Berikut merupakan perbandingan hasil akurasi pada setiap model sebelum dilakukan resampling :

	Akurasi Tanpa Resampling
Random Forest	0.80
Logistic Regression	0.77
Decision Tree	0.72

SVM	0.77
KNN	0.78

Tabel 1. Perbandingan akurasi tiap model tanpa resampling

Pada algoritma Random Forest tanpa resampling mendapatkan akurasi sebanyak 0.79, Logistik Regression 0.77, Decision Tree 0.72, SVM 0.77, KNN 0.78. Sehingga dibandingkan dengan sebelumnya, performa model yang paling baik yakni terdapat pada model random forest setelah dilakukan resampling, dengan akurasi model yang mencapai 0.95.

Prediksi

1. Preprocessing Dataset BMKG

- Memilih kolom yang relevan dengan data train model dan melakukan label encoder

	latitude	longitude	magnitude	magType	depth	nst	gap
0	-1.710596	120.091766	2.837047	M	10	23	76.859665
1	-5.944668	130.647629	4.846796	Mlv	133	48	78.901794
2	-6.438559	107.307701	2.836673	M	10	25	80.936186
3	-7.134665	107.558365	4.171044	M	10	69	83.403366
4	-7.266338	129.565552	4.551028	M	166	68	55.587601
...
661	-8.737351	109.740349	3.031500	M	12	22	237.116982
662	4.079156	128.337433	4.160199	M	14	19	255.591034
663	-0.221772	123.710251	3.289224	M	54	27	87.837761
664	-5.309146	102.562805	4.191902	M	13	29	156.701263
665	-5.751341	112.520355	4.559490	M	10	64	87.903351

666 rows x 7 columns

	latitude	longitude	magnitude	magType	depth	nst	gap
0	-1.710596	120.091766	2.837047	0	10	23	76.859665
1	-5.944668	130.647629	4.846796	1	133	48	78.901794
2	-6.438559	107.307701	2.836673	0	10	25	80.936186
3	-7.134665	107.558365	4.171044	0	10	69	83.403366
4	-7.266338	129.565552	4.551028	0	166	68	55.587601
...
661	-8.737351	109.740349	3.031500	0	12	22	237.116982
662	4.079156	128.337433	4.160199	0	14	19	255.591034
663	-0.221772	123.710251	3.289224	0	54	27	87.837761
664	-5.309146	102.562805	4.191902	0	13	29	156.701263
665	-5.751341	112.520355	4.559490	0	10	64	87.903351

666 rows x 7 columns

- Standarisasi data

	latitude	longitude	magnitude	magType	depth	nst	gap
0	-0.095105	0.659371	-10.505247	-7.978923	-0.760171	-0.171864	3.975041
1	-0.241630	0.743235	-5.306645	-5.938718	1.283796	0.009529	4.129445
2	-0.258722	0.557803	-10.506214	-7.978923	-0.760171	-0.157352	4.283264
3	-0.282811	0.559795	-7.054608	-7.978923	-0.760171	0.161898	4.469806
4	-0.287368	0.734638	-6.071705	-7.978923	1.832177	0.154642	2.366674
...
661	-0.338274	0.577130	-10.002258	-7.978923	-0.726936	-0.179119	16.091994
662	0.105257	0.724881	-7.082659	-7.978923	-0.693701	-0.200886	17.488806
663	-0.043582	0.688119	-9.335605	-7.978923	-0.028996	-0.142841	4.805088
664	-0.219637	0.520106	-7.000655	-7.978923	-0.710318	-0.128330	10.011813
665	-0.234940	0.599217	-6.049815	-7.978923	-0.760171	0.125620	4.810047

666 rows x 7 columns

Gambar 14. Menggunakan model standarisasi yang disimpan

Melakukan perbandingan standarisasi standard scaller yang berbeda mean dan standar deviasi untuk melihat perbedaan hasil prediksi yang dihasilkan. Standarisasi yang pertama yakni menggunakan model standarisasi yang sama dengan standarisasi yang dilakukan saat train model dan telah disimpan sebelumnya saat melakukan train model. Standarisasi yang kedua yakni melakukan fit_transform lagi ke dataset bmkg sehingga didapatkan hasil standarisasi yang baru atau tidak sama dengan sebelumnya yang menggunakan model standarisasi yang disimpan, yakni bisa dilihat pada gambar berikut :

	latitude	longitude	magnitude	magType	depth	nst	gap
0	0.342213	-0.013796	-0.812283	-0.285642	-0.580228	-0.506506	-0.812661
1	-0.667512	1.001680	2.190371	2.733998	1.637020	0.463288	-0.779027
2	-0.785293	-1.243626	-0.812842	-0.285642	-0.580228	-0.428923	-0.745520
3	-0.951298	-1.219512	1.180767	-0.285642	-0.580228	1.277914	-0.704885
4	-0.982699	0.897584	1.748480	-0.285642	2.231891	1.239123	-1.163019
...
661	-1.333500	-1.009605	-0.521762	-0.285642	-0.544176	-0.545298	1.826823
662	1.722930	0.779439	1.164565	-0.285642	-0.508123	-0.661673	2.131096
663	0.697261	0.334303	-0.136711	-0.285642	0.212934	-0.351339	-0.631849
664	-0.515955	-1.700086	1.211930	-0.285642	-0.526149	-0.273756	0.502353
665	-0.621408	-0.742167	1.761124	-0.285642	-0.580228	1.083956	-0.630769

666 rows x 7 columns

Gambar 15. Standarisasi tanpa model standarisasi yang disimpan (baru)

2. Load Model

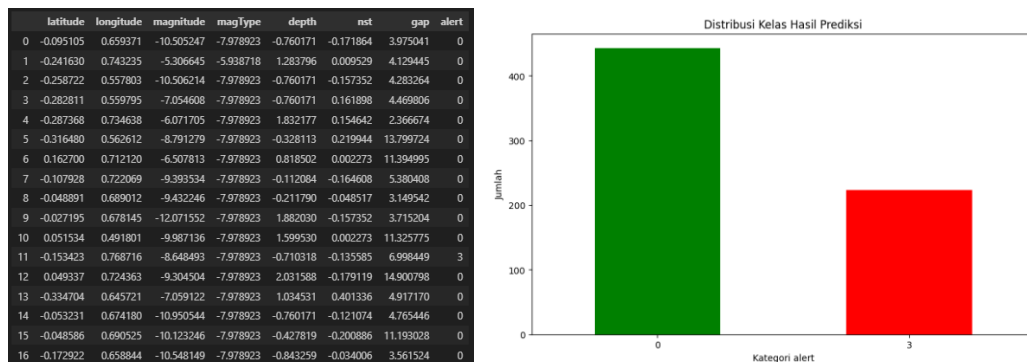
```
import joblib

# Memuat model dari file
load_model = joblib.load('rf2_model.pkl')
```

Menggunakan model yang sebelumnya telah dibuat saat train model, yakni menggunakan model random forest dengan performa yang paling baik diantara model yang lain. Selanjutnya model ini akan digunakan untuk memprediksi label pada dataset BMKG yang masih belum memiliki label alert.

3. Hasil Prediksi

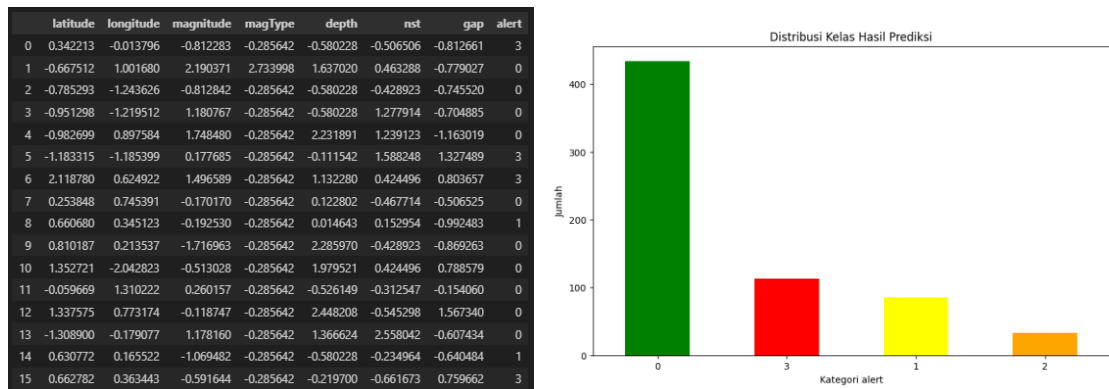
- Hasil prediksi menggunakan standarisasi yang disimpan



Gambar 16. Hasil prediksi standarisasi yang disimpan

Berdasarkan gambar 16 menunjukkan bahwa hasil prediksi menggunakan standarisasi yang disimpan atau standarisasi yang sama dengan model train sebagian besar hanya mampu mengenali atau mendeteksi label 0 dan 3, sedangkan untuk label yang lain tidak ada yang terdeteksi.

- Hasil prediksi tanpa menggunakan standarisasi yang disimpan



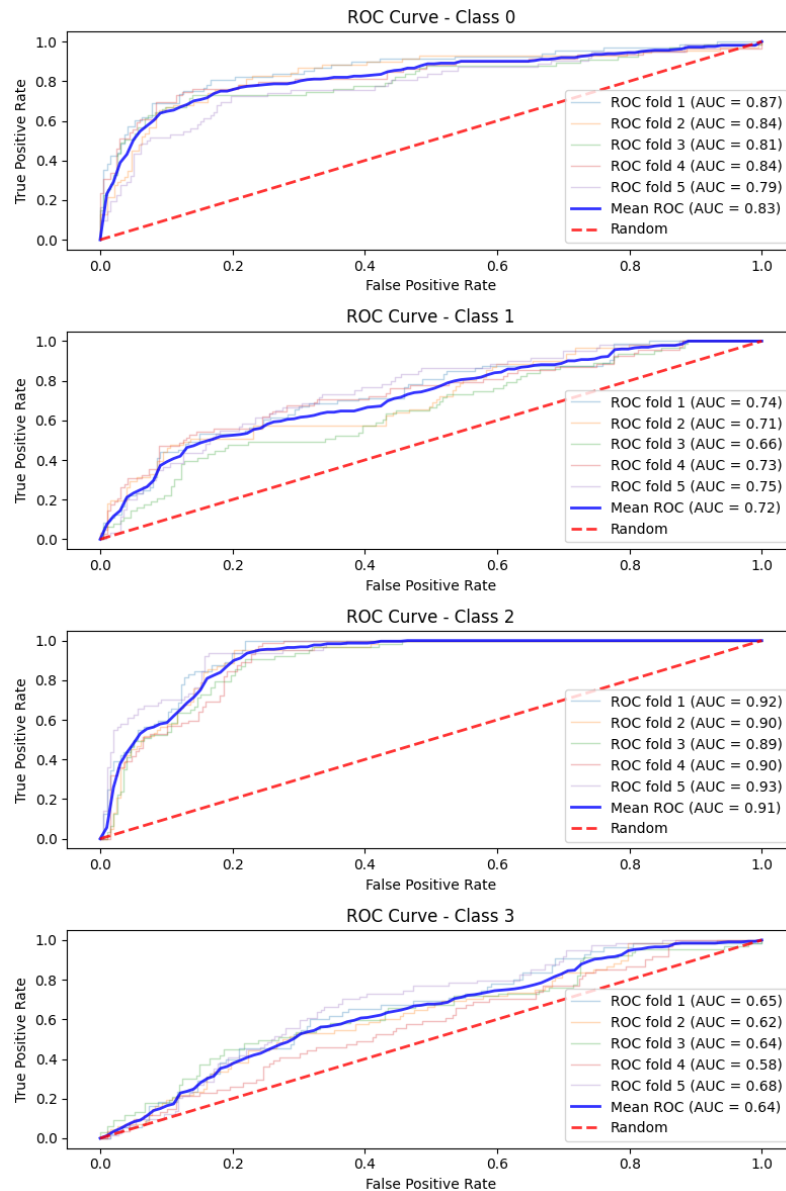
Gambar 17. Hasil prediksi standarisasi yang baru

Berdasarkan gambar 17 menunjukkan bahwa hasil prediksi menggunakan standarisasi yang baru menghasilkan prediksi yang mampu mengenali atau mendeteksi semua kelas baik kelas 0,1,2, dan 3 dibandingkan dengan yang sebelumnya.

TAMBAHAN REVISI

1. ROC dan K-Fold Cross-Validation

Receiver Operating Characteristic (ROC) curve adalah plot grafis yang menggambarkan kemampuan diagnostik dari classifier biner saat ambang diskriminasi bervariasi. Langkah untuk menghasilkan ROC, pada **inisialisasi dan pelatihan model** menggunakan Logistic Regression dalam contoh ini, tetapi juga dapat diterapkan pada classifier apapun. Kemudian pada **K-Fold Cross-Validation** membagi data menjadi set pelatihan dan pengujian beberapa kali (folds) untuk memastikan kinerja model dievaluasi dengan baik. Kemudian tahap **Perhitungan Kurva ROC** untuk setiap fold prediksi probabilitas untuk set uji, menghitung True Positive Rate (TPR) dan False Positive Rate (FPR) pada berbagai ambang batas. Setelahnya dilakukan **Plotting**, Plot TPR terhadap FPR untuk membuat kurva ROC. Area dibawah kurva ROC (AUC) dihitung untuk merangkum kinerja model. Nilai AUC yang lebih tinggi menunjukkan kinerja yang lebih baik.



Gambar menunjukkan kurva ROC untuk empat kelas yang berbeda dalam klasifikasi multi-kelas. Penjelasan:

➤ ROC Curve - Class 0

- **AUC (Area Under the Curve):** Nilai AUC rata-rata (Mean ROC) adalah 0.83, yang menunjukkan bahwa model memiliki kinerja yang baik dalam membedakan kelas 0 dari yang lain. Nilai AUC untuk setiap fold berkisar antara 0.79 hingga 0.87.

- **Interpretasi:** Model cukup andal dalam mengklasifikasikan instance yang termasuk dalam kelas 0. Kurva ROC berada jauh di atas garis diagonal (random), menunjukkan kinerja yang jauh lebih baik daripada tebakan acak.

➤ ROC Curve - Class 1

- **AUC:** Nilai AUC rata-rata adalah 0.72, yang masih menunjukkan kinerja yang lumayan baik namun lebih rendah dibandingkan dengan kelas 0. Nilai AUC untuk setiap fold berkisar antara 0.66 hingga 0.75.

- **Interpretasi:** Model agak baik dalam mengklasifikasikan instance kelas 1, meskipun tidak sebaik dalam kasus kelas 0. Kurva ROC berada di atas garis diagonal, tetapi lebih dekat ke garis acak dibandingkan dengan kurva untuk kelas 0.

➤ ROC Curve - Class 2

- **AUC:** Nilai AUC rata-rata adalah 0.91, menunjukkan kinerja yang sangat baik. Nilai AUC untuk setiap fold berkisar antara 0.89 hingga 0.93.

- **Interpretasi:** Model sangat andal dalam mengklasifikasikan instance yang termasuk dalam kelas 2. Kurva ROC untuk kelas ini berada jauh di atas garis diagonal, menunjukkan kinerja yang sangat baik.

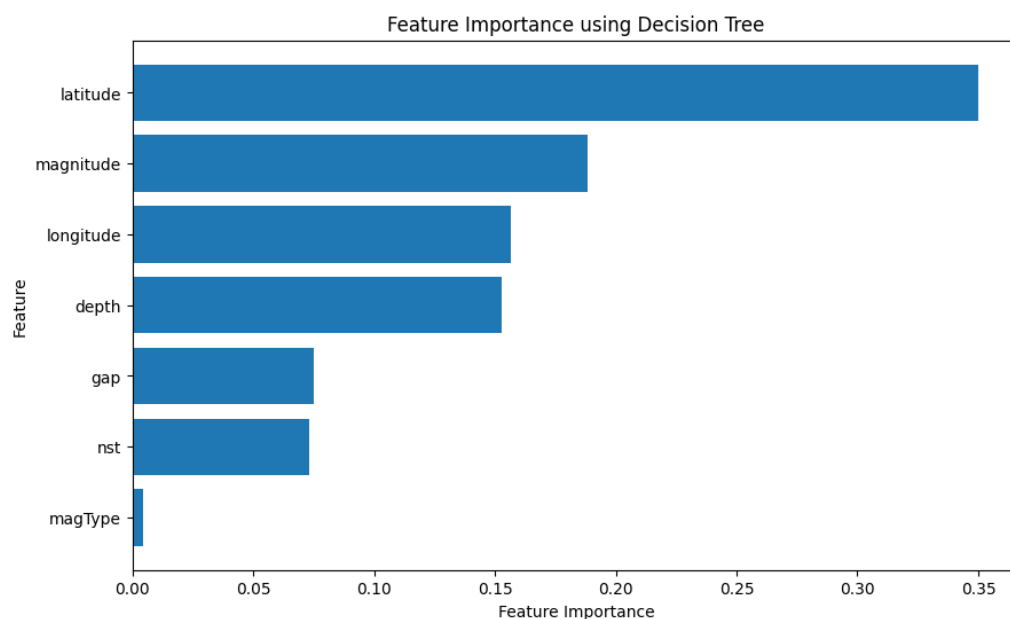
➤ ROC Curve - Class 3

- **AUC:** Nilai AUC rata-rata adalah 0.64, yang menunjukkan kinerja yang moderat. Nilai AUC untuk setiap fold berkisar antara 0.58 hingga 0.68.

- **Interpretasi:** Model memiliki kinerja yang kurang baik dalam mengklasifikasikan instance kelas 3. Kurva ROC lebih mendekati garis diagonal, menunjukkan bahwa kinerja model hampir mendekati tebakan acak.

Berdasarkan hal tersebut, kelas dengan kinerja terbaik adalah kelas 2, kinerja terburuk kelas 3, 0 dan 1 cukup baik.

2. Feature Important



Gambar yang Feature importance adalah ukuran seberapa penting masing-masing fitur dalam memprediksi target variabel. Dalam konteks ini, model Decision Tree telah menentukan seberapa besar kontribusi setiap fitur terhadap prediksi model.

1. Latitude (Sekitar 0.35)

Latitude adalah fitur yang paling penting dalam model ini dengan skor pentingnya sekitar 0.35. Ini berarti latitude memiliki pengaruh terbesar dalam membuat keputusan dalam model Decision Tree.

2. Magnitude (Sekitar 0.20)

Magnitude adalah fitur kedua yang paling penting dengan skor pentingnya sekitar 0.20. Magnitude juga memberikan kontribusi yang signifikan dalam membuat prediksi.

3. Longitude (Sekitar 0.15)

Longitude juga penting, dengan skor sekitar 0.15. Longitude memberikan kontribusi yang berarti dalam model meskipun tidak sebesar latitude atau magnitude.

4. Depth (Sekitar 0.15)

Depth memiliki skor penting yang hampir sama dengan longitude. Ini menunjukkan bahwa kedalaman juga memainkan peran penting dalam prediksi.

5. Gap (Sekitar 0.10)

Gap memiliki skor penting sekitar 0.10, menunjukkan bahwa meskipun tidak sepenting latitude atau magnitude, fitur ini masih memberikan kontribusi yang cukup besar.

6. NST (Sekitar 0.05)

NST memiliki skor penting yang lebih rendah dibandingkan fitur-fitur lainnya, tetapi masih memberikan kontribusi dalam model.

7. MagType (Sangat Kecil)

MagType memiliki skor penting yang sangat kecil, hampir mendekati 0. Ini berarti fitur ini memiliki kontribusi yang sangat kecil atau hampir tidak ada dalam membuat prediksi.

Jadi Latitude dan magnitude adalah fitur utama yang paling berpengaruh dalam model Decision Tree ini. Longitude dan depth juga penting, meskipun tidak sebesar latitude atau magnitude. Gap dan NST memberikan kontribusi yang lebih kecil, tetapi masih relevan.

MagType hampir tidak memberikan kontribusi dalam prediksi, karena dia merupakan fitur kategori.

Kesimpulan

Project ini bertujuan untuk memprediksi tingkat keparahan gempa bumi di Indonesia menggunakan beberapa model machine learning. Setelah melakukan build model, hasil akurasi tiap model menunjukkan perbedaan yang signifikan antara model tanpa resampling dan dengan resampling. Secara keseluruhan, resampling meningkatkan akurasi sebagian besar model, dengan Random Forest sebagai model terbaik setelah resampling, meningkat dari 80% menjadi 95%. Decision Tree, SVM, dan KNN juga mengalami peningkatan akurasi yang substansial. Sebaliknya, Logistic Regression mengalami penurunan akurasi setelah resampling. Hasil ini menekankan pentingnya pemilihan dan pengoptimalan model dengan teknik resampling untuk meningkatkan akurasi prediksi tingkat keparahan gempa bumi. Dalam proyek ini, kami mengembangkan model machine learning untuk memprediksi tingkat keparahan gempa bumi di Indonesia, menggunakan data spasial dari dua sumber: dataset gempa bumi dari Kaggle dan dataset dari BMKG. Setelah tahap preprocessing seperti memeriksa dan menghapus missing value, menangani outliers, dan standarisasi data, kami menggunakan teknik resampling SMOTE untuk mengatasi ketidakseimbangan kelas. Kami membangun beberapa model machine learning, termasuk Random Forest, Logistic Regression, Decision Tree, SVM, dan KNN, dan menemukan bahwa model Random Forest dengan resampling memberikan hasil prediksi yang paling akurat dengan akurasi mencapai 95%.

Analisis lebih lanjut menunjukkan bahwa latitude dan magnitude adalah fitur paling penting dalam memprediksi tingkat keparahan gempa bumi, diikuti oleh longitude dan depth. Teknik resampling secara signifikan meningkatkan akurasi sebagian besar model, kecuali Logistic Regression yang mengalami penurunan. Kinerja model juga dievaluasi menggunakan ROC curve dan K-Fold Cross-Validation, yang menunjukkan bahwa model memiliki kinerja terbaik dalam mengklasifikasikan kelas tertentu, terutama kelas 2. Dengan menggunakan model yang dilatih, kami mampu memberikan prediksi yang lebih akurat untuk dataset BMKG.

Proyek ini menghasilkan pemahaman mendalam tentang penggunaan data spasial dan teknik machine learning untuk mitigasi bencana gempa bumi. Model prediksi yang dikembangkan dapat membantu pemerintah dan badan terkait dalam meningkatkan kesiapsiagaan dan perencanaan infrastruktur yang lebih tahan gempa. Selain itu, informasi penting yang dihasilkan dari model ini dapat meningkatkan kesadaran dan kesiapsiagaan masyarakat terhadap potensi gempa bumi di Indonesia, serta mendukung pembangunan yang lebih aman dan berkelanjutan.