

## > Klasifikasi Citra Satelit dan Visualisasi Spasial

↳ 1 cell hidden

### ▼ 1. Klasifikasi Spasial

```
# Import Libraries
import geopandas as gpd
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Baca Data
shp_file = '/content/processed_jkt.shp'
shp_file = gpd.read_file(shp_file)
shp_file.head()
```

Show hidden output

```
# Cek informasi dari shp_file
shp_file.info()
print(shp_file.crs)
print(shp_file.shape)
print(shp_file.describe())
```

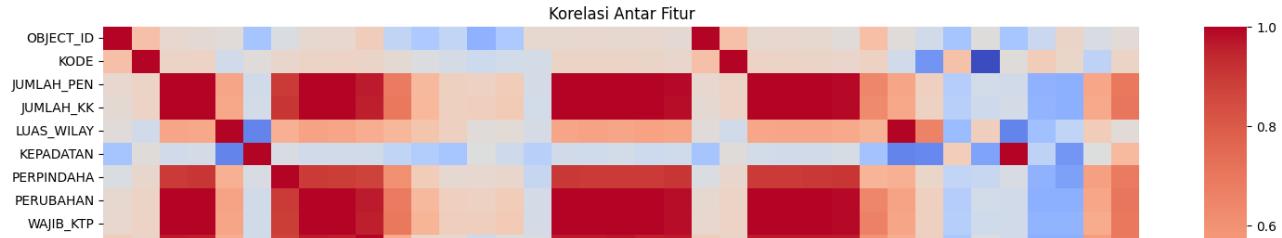
Show hidden output

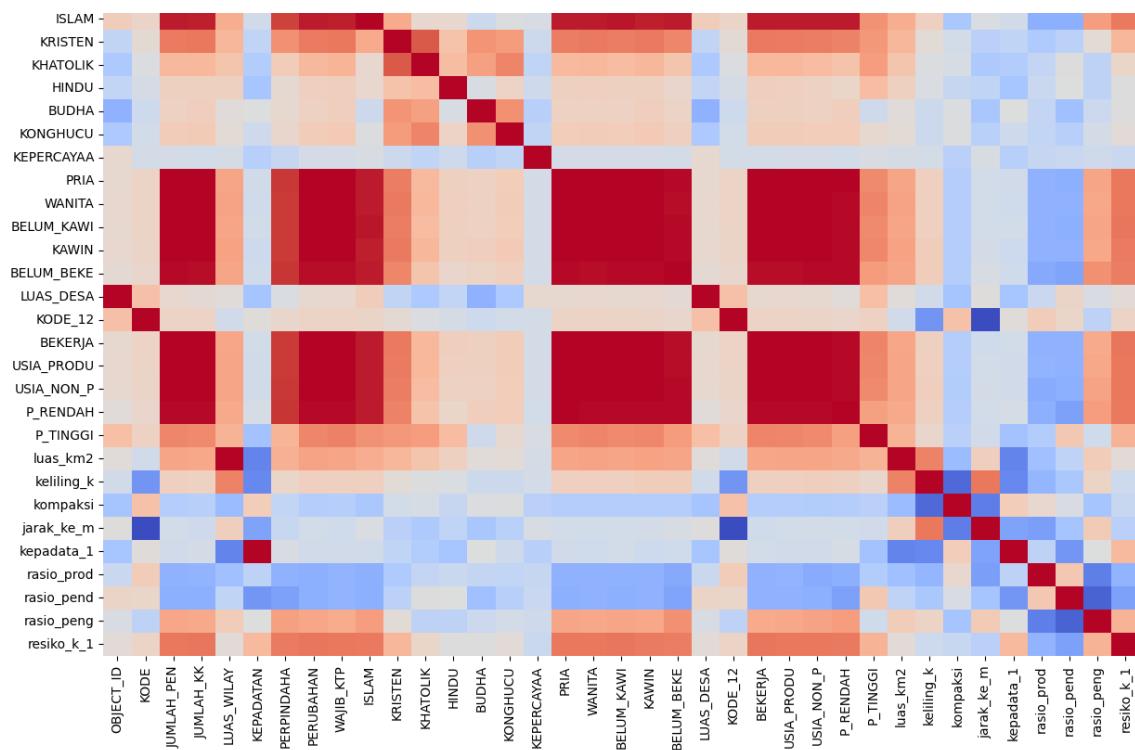
```
# Cek Korelasi Antar Fitur/Kolom
num_data = shp_file.select_dtypes(include=[np.number])
corr = num_data.corr()
corr
```

Show hidden output

```
# Visualisasikan dengan Heatmap
plt.figure(figsize=(18, 12))
sns.heatmap(corr, cmap='coolwarm', annot=False, fmt='.2f')
plt.title('Korelasi Antar Fitur')
plt.show()
```

Show hidden output





```
# Feature Selection
X = shp_file.drop(['KODE_DESA', 'DESA', 'PROVINSI', 'KAB_KOTA', 'KECAMATAN',
                   'DESA_KELUR', 'GENERATED', 'KODE_DES_1', 'KODE_DES_3',
                   'DESA_KEL_1', 'resiko_kre', 'resiko_k_1', 'geometry'], axis=1)
y = shp_file['resiko_k_1']

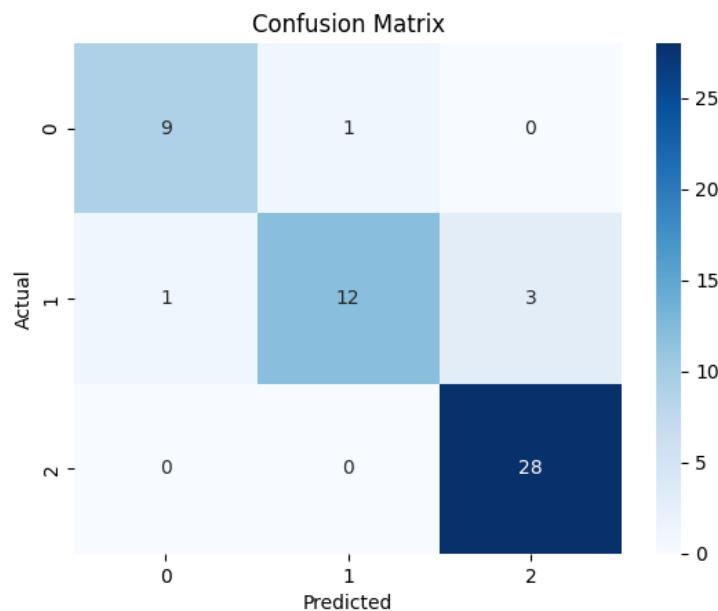
# Membagi data menjadi X Train, y Train
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Membangun model random forest dan evaluasi model
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
print(classification_report(y_test, y_pred_rf))
```

	precision	recall	f1-score	support
0	0.90	0.90	0.90	10
1	0.92	0.75	0.83	16
2	0.90	1.00	0.95	28
accuracy			0.91	54
macro avg	0.91	0.88	0.89	54
weighted avg	0.91	0.91	0.90	54

```
# Visualisasi dengan menggunakan confussion matrix
cm = confusion_matrix(y_test, y_pred_rf)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt
```

```
→ <module 'matplotlib.pyplot' from '/usr/local/lib/python3.11/dist-packages/matplotlib/pyplot.py'>
```



+ Code

+ Text

```
# Analisis Feature Importance
feature_importance = rf.feature_importances_
feature_names = X.columns
feature_importance_df = pd.DataFrame({'Feature': feature_names, 'Importance': feature_importance})
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)
```

```
print(feature_importance_df)

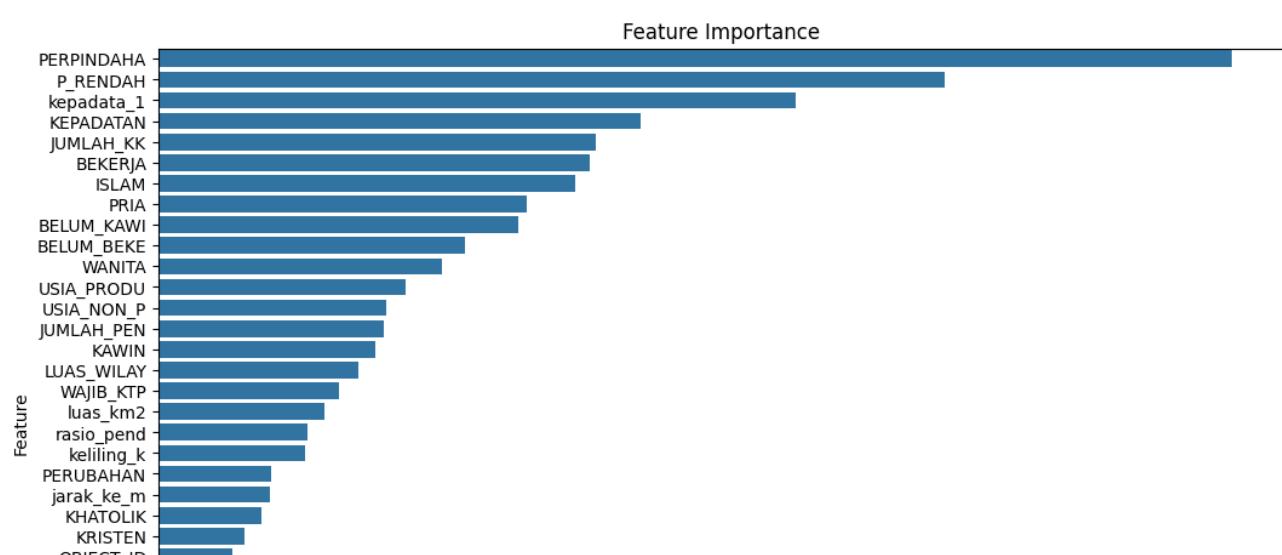
Feature Importance
6 PERPINDAH 0.127225
26 P_RENDAH 0.093172
32 kepadata_1 0.075601
5 KEPADATAN 0.057206
3 JUMLAH_KK 0.051907
23 BEKERJA 0.051217
9 ISLAM 0.049460
16 PRIA 0.043687
18 BELUM_KAWI 0.042715
20 BELUM_BEKE 0.036315
17 WANITA 0.033633
24 USIA_PRODU 0.029321
25 USIA_NON_P 0.026968
2 JUMLAH_PEN 0.026771
19 KAWIN 0.025673
4 LUAS_WILAY 0.023693
8 WAJIB_KTP 0.021475
28 luas_km2 0.019759
34 rasio_pend 0.017712
29 keliling_k 0.017439
7 PERUBAHAN 0.013330
31 jarak_ke_m 0.013290
11 KHATOLIK 0.012166
10 KRISTEN 0.010181
0 OBJECT_ID 0.008814
35 rasio_peng 0.008432
27 P_TINGGI 0.008102
30 kompaksi 0.007908
33 rasio_prod 0.007276
21 LUAS_DESA 0.007276
1 KODE 0.006651
14 KONGHUCU 0.006488
```

[https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6iBfoQghudFi#scrollTo=oqFlkpfHdn\\_n](https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6iBfoQghudFi#scrollTo=oqFlkpfHdn_n)

Page 7 of 34

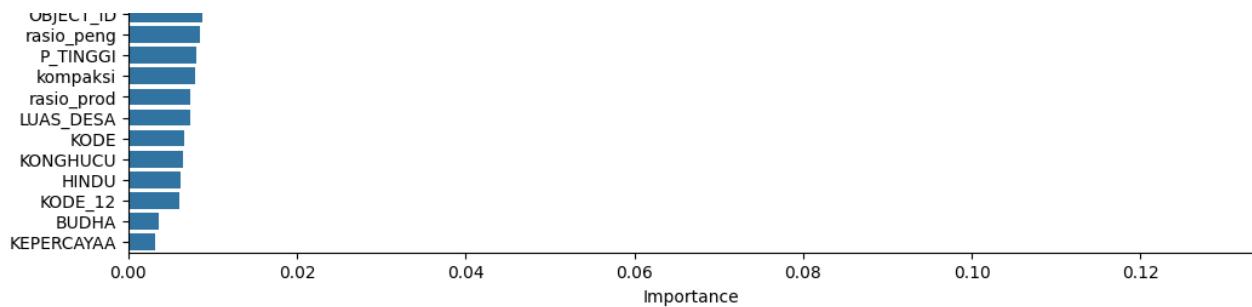
```
12 HINDU 0.006151
22 KODE_12 0.006095
13 BUDHA 0.003673
15 KEPERCAYAA 0.003218
```

```
# Visualisasi Feature Importance
plt.figure(figsize=(12,8))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df)
plt.title('Feature Importance')
plt.show()
```



[https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6iBfoQghudFi#scrollTo=oqFlkpfHdn\\_n](https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6iBfoQghudFi#scrollTo=oqFlkpfHdn_n)

Page 8 of 34



```
# Hyperparameter Tuning
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
    'bootstrap': [True, False]
}

# Inisialisasi GridSearchCV dengan RandomForestClassifier sebagai estimator
grid_search = GridSearchCV(estimator=RandomForestClassifier(random_state=42), # Model dasar yang akan dituning
                           param_grid=param_grid, # Parameter yang akan dicoba satu per satu
                           cv=5, # Cross-validation sebanyak 5 fold
                           scoring='accuracy', # Gunakan akurasi sebagai metrik evaluasi
                           n_jobs=-1, # Gunakan semua core CPU untuk paralel proses
                           verbose=1) # Tampilkan progres selama proses training

grid_search.fit(X_train, y_train)
```

→ Fitting 5 folds for each of 48 candidates, totalling 240 fits

- ▶ **GridSearchCV** (i) (?)
- ▶ **best\_estimator\_:** **RandomForestClassifier**
  - ▶ **RandomForestClassifier** (?)

```
# Evaluasi model setelah dilakukan HT
print('Best Parameter:', grid_search.best_params_)
print('Best Score:', grid_search.best_score_)

→ Best Parameter: {'bootstrap': True, 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}
Best Score: 0.8825027685492802
```

```
# Simpan Best Model
best_model = grid_search.best_estimator_ # Simpan model terbaik berdasarkan evaluasi Grid Search

y_pred_best = best_model.predict(X_test) # Gunakan model terbaik untuk prediksi data uji

shp_file['prediksi_rf'] = best_model.predict(X)
# Tampilkan hasil evaluasi model terbaik
print("\nEvaluasi Model Setelah Grid Search:")
print(classification_report(y_test, y_pred_best)) # Tampilkan metrik evaluasi: precision, recall, f1-score, dll
```

```
→ Evaluasi Model Setelah Grid Search:
      precision    recall   f1-score   support
0          0.90     0.90     0.90      10
1          0.92     0.75     0.83      16
2          0.90     1.00     0.95      28

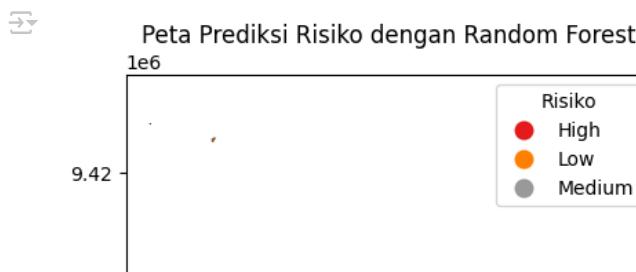
accuracy                           0.91      54
macro avg       0.91     0.88     0.89      54
weighted avg    0.91     0.91     0.90      54
```

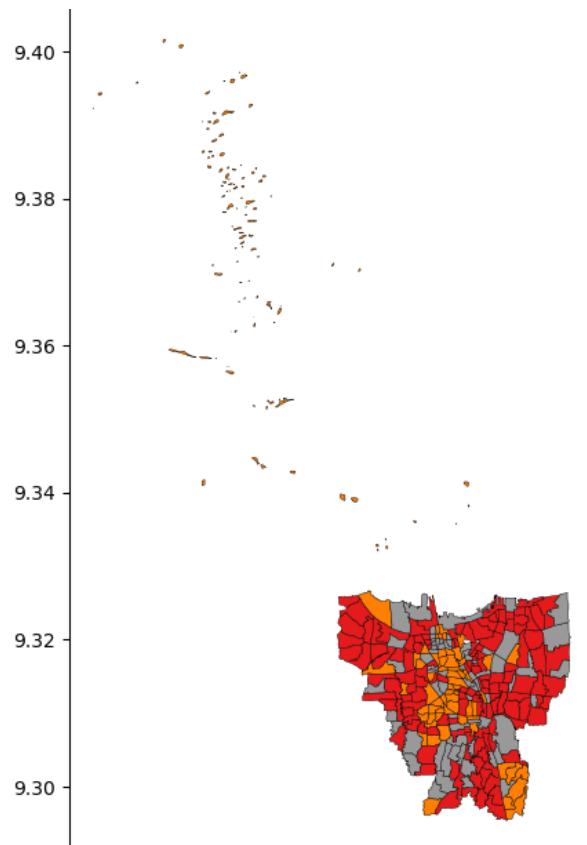
```
# Visualisasi hasil Prediksi Random Forest

label_map = {0: 'Low', 1: 'Medium', 2: 'High'}
shp_file['prediksi_label'] = shp_file['prediksi_rf'].map(label_map)
fig, ax = plt.subplots(1, 1, figsize=(12, 10))

shp_file.plot(
    column='prediksi_label',
    categorical=True,
    legend=True,
    legend_kwds={'title': 'Risiko'},
    cmap='Set1',
    ax=ax,
    edgecolor='black',
    linewidth=0.3
)

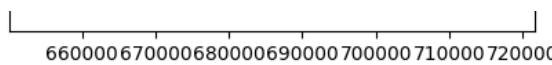
ax.set_title('Peta Prediksi Risiko dengan Random Forest')
ax.axis('on')
plt.show()
```





[https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6lBfoQghudFi#scrollTo=oqFlkpfHdn\\_n](https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6lBfoQghudFi#scrollTo=oqFlkpfHdn_n)

Page 13 of 34



```
# Simpan dalam file baru
output = '/content/processed_jkt_rf.shp'
shp_file.to_file(output)
```

## 2. Segmentasi Citra

!pip install rasterio

Show hidden output

```
import rasterio
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import matplotlib.colors as mcolors
import matplotlib.patches as mpatches
```

```
# Baca Data Raster
file = '/content/landsat8_dki_jakpus_2017.tif'
with rasterio.open(file) as src:
    band_data = src.read([2, 3, 4])
```

[https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6lBfoQghudFi#scrollTo=oqFlkpfHdn\\_n](https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6lBfoQghudFi#scrollTo=oqFlkpfHdn_n)

Page 14 of 34

```
# Cek informasi data
bands, height, width = band_data.shape
print(band_data.shape)
print(band_data.dtype)

→ (3, 349, 334)
float64

# Pre Processing Data
blue = band_data[0].astype(float)
green = band_data[1].astype(float)
red = band_data[2].astype(float)

# Bersihkan data
# Ganti nilai NaN (kosong) menjadi 0 untuk menghindai eror
for band in [red, green, blue]:
    band[np.isnan(band)] = 0

# Ubah citra 3D menjadi 2D dengan Transform
image_reshaped = band_data.reshape(bands, height * width).T

# Pilih Valid Pixel
valid_pixels = ~np.isnan(image_reshaped).any(axis=1)
image_valid = image_reshaped[valid_pixels]
```

```
# Normalisasi Data
# Hitung nilai minimum dan maksimum untuk setiap band (R, G, B)
min_val = image_valid.min(axis=0)
max_val = image_valid.max(axis=0)

# Hitung rentang nilai (range = max - min)
range_val = max_val - min_val

# Jika ada band yang range-nya nol (semua nilainya sama), ubah jadi 1 agar tidak dibagi 0
range_val[range_val == 0] = 1

# Lakukan normalisasi min-max ke rentang [0, 1]
image_normalized = (image_valid - min_val) / range_val

from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.utils import resample
import matplotlib.pyplot as plt
```

```
# Buat Sample Acak untuk uji SSE dan SIL
sample = resample(image_normalized, n_samples=10000, random_state=42)

# Inisialisasi list kosong untuk menyimpan hasil evaluasi
sse = []
sil = []

# Uji jumlah cluster dari 2 sampai 6
for k in range(2, 6):
    model = KMeans(
        n_clusters=k,
        random_state=42
    ).fit(image_normalized)

    # Simpan nilai SSE (semakin kecil nilainya biasanya semakin baik)
    sse.append(model.inertia_)

    # Prediksi label cluster untuk data sample
    labels = model.predict(sample)

    # Hitung dan simpan Silhouette Score (semakin besar nilainya semakin baik)
    sil.append(silhouette_score(sample, labels))

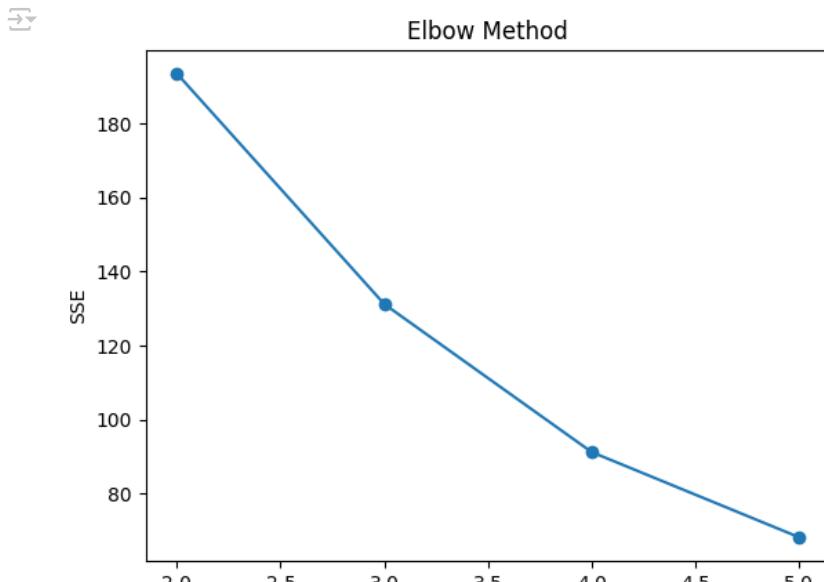
# Visualisasi SSE
plt.figure()
plt.plot(range(2, 6), sse, marker='o')
plt.title('Elbow Method')
plt.xlabel('Jumlah Cluster (k)')
plt.ylabel('SSE')
```

[https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6iBfoQghudFi#scrollTo=oqFlkpfHdn\\_n](https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6iBfoQghudFi#scrollTo=oqFlkpfHdn_n)

Page 17 of 34

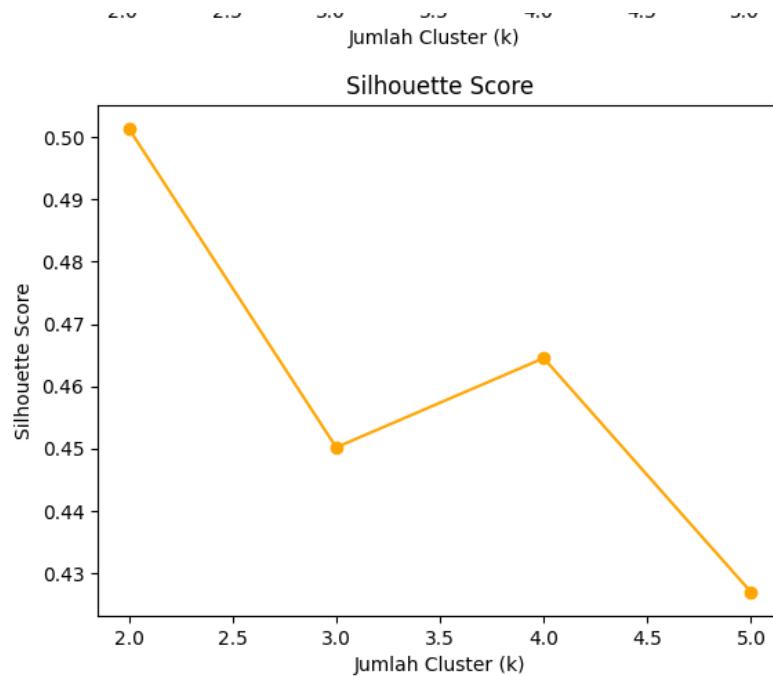
```
plt.show()

# Visualisasi Silhouette Score
plt.figure()
plt.plot(range(2, 6), sil, marker='o', color='orange')
plt.title('Silhouette Score')
plt.xlabel('Jumlah Cluster (k)')
plt.ylabel('Silhouette Score')
plt.show()
```



[https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6iBfoQghudFi#scrollTo=oqFlkpfHdn\\_n](https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6iBfoQghudFi#scrollTo=oqFlkpfHdn_n)

Page 18 of 34



```
# Membangun model K-Means
jumlah_cluster = 4 # Jumlah klaster sesuai dengan hasil SSE/SIL

kmeans = KMeans(
    n_clusters=jumlah_cluster,
    random_state=42
)

cluster_labels = kmeans.fit_predict(image_normalized)

all_labels = -1 * np.ones(image_reshaped.shape[0], dtype=int)
all_labels[valid_pixels] = cluster_labels
clustered_image = all_labels.reshape(height, width)

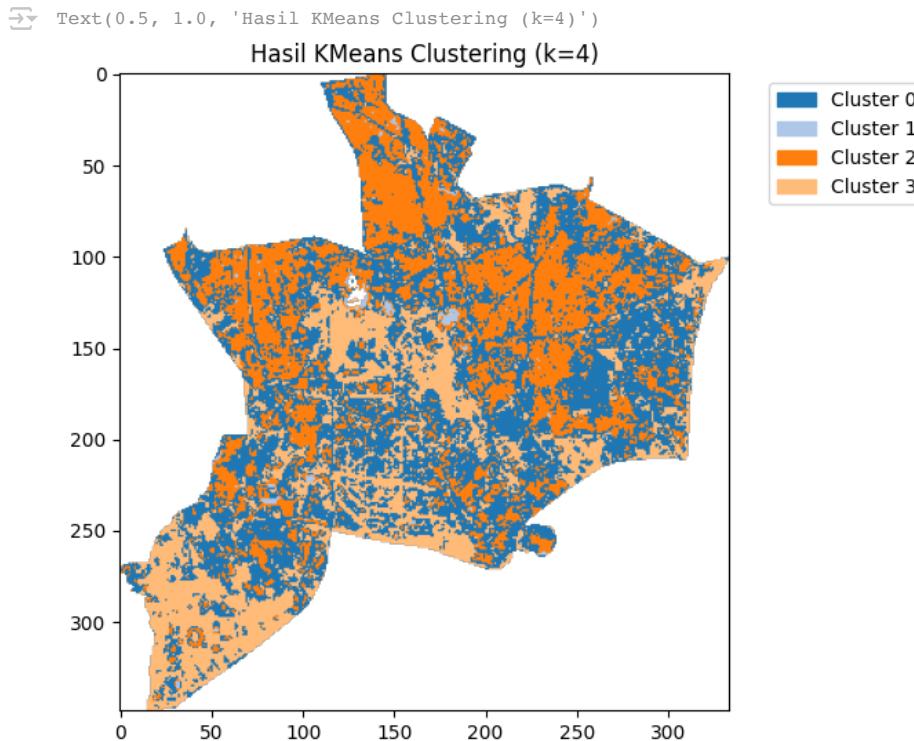
# Visualisasi Hasil Segmentasi
masked_image = np.where(clustered_image == -1, np.nan, clustered_image)
cmap = plt.cm.tab20
custom_cmap = mcolors.ListedColormap(cmap.colors[:jumlah_cluster])

plt.figure(figsize=(8, 6))
plt.imshow(masked_image, cmap=custom_cmap)
legend_labels = [f'Cluster {i}' for i in range(jumlah_cluster)]
legend_patches = [mpatches.Patch(color=custom_cmap.colors[i], label=legend_labels[i])
                  for i in range(jumlah_cluster)]

# Tampilkan legend di luar plot sebelah kanan atas
plt.legend(handles=legend_patches, bbox_to_anchor=(1.05, 1), loc='upper left')

plt.axis('on')
```

```
plt.title(f'Hasil KMeans Clustering (k={jumlah_cluster})')
```



```
# Simpan hasil segmentasi
output = '/content/landsat8_dki_jakpus_2017_cluster.tif'
with rasterio.open(file) as src:
    meta = src.meta.copy()
    meta.update({'count': 1,
                 'dtype': 'int32',
                 'nodata': -1})
    dst = rasterio.open(output, 'w', **meta)
    dst.write(clustered_image.astype('int32'), 1)
```

## ▼ 3. Visualisasi Interaktif

### ▼ 3.1 Folium Vektor

```
import folium
```

```
# Baca Data
data = gpd.read_file('/content/processed_jkt_rf.shp')
print(data.columns)

→ Index(['OBJECT_ID', 'KODE_DESA', 'DESA', 'KODE', 'PROVINSI', 'KAB_KOTA',
       'KECAMATAN', 'DESA_KELUR', 'JUMLAH_PEN', 'JUMLAH_KK', 'LUAS_WILAY',
       'KEPADATAN', 'PERPINDAHAN', 'PERUBAHAN', 'WAJIB_KTP', 'ISLAM', 'KRISTEN',
       'KHATOLIK', 'HINDU', 'BUDHA', 'KONGHUCU', 'KEPERCAYAA', 'PRIA',
       'WANITA', 'BELUM_KAWI', 'KAWIN', 'BELUM_BEKE', 'GENERATED',
       'KODE_DES_1', 'LUAS_DESA', 'KODE_DES_3', 'DESA_KEL_1', 'KODE_12',
       'BEKERJA', 'USIA_PRODU', 'USIA_NON_P', 'P_RENDERAH', 'P_TINGGI',
       'resiko_kre', 'luas_km2', 'keliling_k', 'kompaksi', 'jarak_ke_m',
       'kepadata_1', 'rasio_prod', 'rasio_pend', 'rasio_peng', 'resiko_k_1',
       'prediksi_r', 'prediksi_l', 'geometry'],
      dtype='object')

# Konversi nilai prediksi ke label kategori
label_map = {0: 'Low', 1: 'Medium', 2: 'High'}
data['prediksi_label'] = data['prediksi_r'].map(label_map)

# Mengatur warna untuk tiap kategori
color_dict = {'Low': 'green', 'Medium': 'orange', 'High': 'red'}
```

```
# Mengatur tampilan tiap fitur (poligon desa)
def style_function(feature):
    risk = feature['properties']['prediksi_label']
    return {
        'fillColor': color_dict.get(risk, 'gray'),
        'color': 'black',
        'weight': 0.5,
        'fillOpacity': 0.6}

# Membuat peta kosong
m = folium.Map(location=[-6.2, 106.8], zoom_start=10, tiles=None)

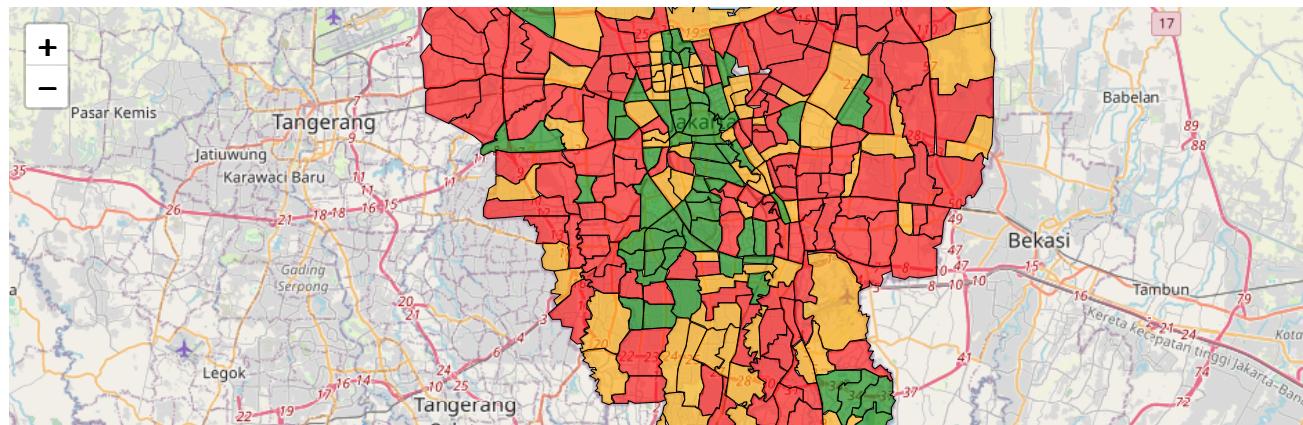
# Membuat Basemap
#Tambahkan beberapa pilihan basemap agar pengguna bisa memilih
folium.TileLayer('CartoDB positron', attr='Map tiles by CartoDB, under CC BY 3.0').add_to(m)
# Menambahkan layer dasar peta 'CartoDB Positron' (tema terang) dengan atribusi lisensi
folium.TileLayer('CartoDB dark_matter', attr='Map tiles by CartoDB, under CC BY 3.0').add_to(m)
# Menambahkan layer dasar peta 'CartoDB Dark Matter' (tema gelap) dengan atribusi lisensi
folium.TileLayer('OpenStreetMap').add_to(m)
# Menambahkan layer dasar peta OpenStreetMap (peta standar open-source)

→ <folium.raster_layers.TileLayer at 0x7a88f4a32bd0>
```

```
# Menambahkan layer GeoJSON dari shapefile
folium.GeoJson(
    data, #Data SHP
    name='Prediksi Risiko', # Nama layer untuk ditampilkan di kontrol layer
    style_function=style_function, # Menggunakan fungsi style_function untuk mengatur tampilan poligon
    tooltip=folium.GeoJsonTooltip(
        fields=['DESA', 'prediksi_label'], # Kolom yang akan ditampilkan di tooltip (nama desa dan kategori risiko
        aliases=['Desa:', 'Risiko:'] # Label alias untuk kolom di tooltip agar lebih mudah dibaca
    )
).add_to(m)
```

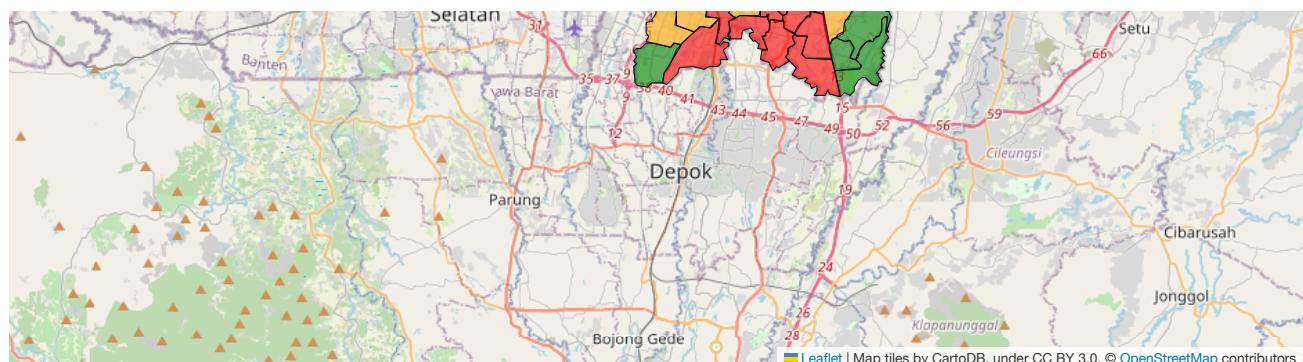
<folium.features.GeoJson at 0x7a88f5c8b6d0>

m



[https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6lBfoQghudFi#scrollTo=oqFlkpfHdn\\_n](https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6lBfoQghudFi#scrollTo=oqFlkpfHdn_n)

Page 25 of 34



```
# Menambahkan Legend
from branca.element import Template, MacroElement

legend_html = """
{%
macro html(this, kwargs) %}
```

[https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6lBfoQghudFi#scrollTo=oqFlkpfHdn\\_n](https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6lBfoQghudFi#scrollTo=oqFlkpfHdn_n)

Page 26 of 34

```

<!-- North Arrow (kompas) -->
<div style="position: fixed; bottom: 10px; right: 10px; z-index: 9999;">
<br>

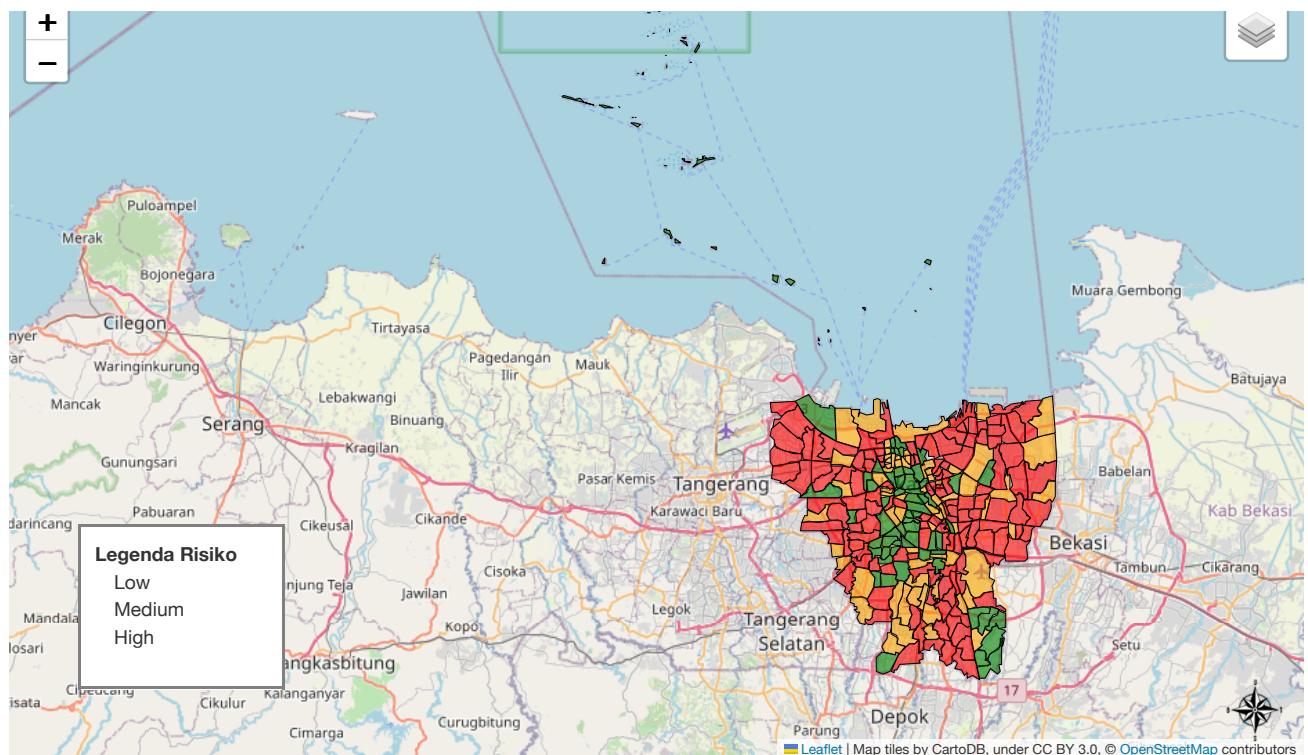
</div>

<!-- Legend Box -->
<div style="position: fixed; bottom: 50px; left: 50px; width: 150px; height: 120px; border: 2px solid grey; z-index: 9999; font-size: 14px; background-color: white; padding: 10px;">
<br>
<b>Legenda Risiko</b><br>
<i style="background-color: green; width: 10px; height: 10px; display: inline-block;"></i>&ampnbspLow<br>
<i style="background-color: orange; width: 10px; height: 10px; display: inline-block;"></i>&ampnbspMedium<br>
<i style="background-color: red; width: 10px; height: 10px; display: inline-block;"></i>&ampnbspHigh<br>
</div>

{%
  endmacro
  .....
}

legend = MacroElement()
legend._template = Template(legend_html)
m.get_root().add_child(legend)

```



```
# Menambahkan kontrol layer agar pengguna bisa pilih basemap dan simpan hasil
folium.LayerControl().add_to(m)

m.save('peta_interaktif_prediksi_rf.html')
```

## ▼ 3.2 Folium Raster

```
from matplotlib import colors
from PIL import Image

# Buka Data
data = '/content/landsat8_dki_jakpus_2017_cluster.tif'
with rasterio.open(data) as src:
    kmeans_array = src.read(1)
    bounds = src.bounds
    nodata_val = src.nodata if src.nodata is not None else 0
```

[https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6iBfoQghudFi#scrollTo=oqFlkpfHdn\\_n](https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6iBfoQghudFi#scrollTo=oqFlkpfHdn_n)

Page 29 of 34

```
# Cek Unique Value
unique_classes = np.unique(kmeans_array[kmeans_array != nodata_val])
print("Kelas unik:", unique_classes)

 Kelas unik: [0 1 2 3]

# Membuat warna label sesuai dengan jumlah kelas
colors_list = ['yellow', 'black', 'brown', 'green']
cmap = colors.ListedColormap(colors_list)
boundaries = [-0.5 + i for i in range(len(unique_classes) + 1)]
norm = colors.BoundaryNorm(boundaries, ncolors=len(colors_list))

# Ubah ke RGBA (dengan transparansi)
rgba_img = cmap(norm(kmeans_array)) # Mengonversi array K-Means ke format RGBA menggunakan colormap dan normalisasi
mask = (kmeans_array != nodata_val) # Membuat mask boolean: True untuk data valid, False untuk nodata
rgba_img[..., 3] = mask.astype(float) * 0.8

# Simpan ke PNG
rgba_img_255 = (rgba_img * 255).astype(np.uint8) # Mengonversi nilai RGBA (0-1) ke rentang 0-255 untuk format gambar
image_pil = Image.fromarray(rgba_img_255, mode='RGBA') # Membuat objek gambar PIL dari array RGBA
image_pil.save('hasil_kmeans.png')

# Hitung bounds untuk overlay
bounds_latlon = [[bounds.bottom, bounds.left], [bounds.top, bounds.right]]
```

[https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6iBfoQghudFi#scrollTo=oqFlkpfHdn\\_n](https://colab.research.google.com/drive/1zKb22edKKwqt1WWvXV8x6iBfoQghudFi#scrollTo=oqFlkpfHdn_n)

Page 30 of 34

```
# Buat peta Folium
m = folium.Map(
    location=[(bounds.top + bounds.bottom) / 2, (bounds.left + bounds.right) / 2], # Menentukan pusat peta berdasarkan titik tengah
    zoom_start=10 # Mengatur level zoom awal peta
)

# Menambahkan base maps
folium.TileLayer('CartoDB positron', attr='Map tiles by CartoDB, under CC BY 3.0').add_to(m) # Menambahkan layer peta positron
folium.TileLayer('CartoDB dark_matter', attr='Map tiles by CartoDB, under CC BY 3.0').add_to(m) # Menambahkan layer peta dark matter
folium.TileLayer('OpenStreetMap').add_to(m) # Menambahkan layer peta dasar OpenStreetMap

↳ <folium.raster_layers.TileLayer at 0x7a88f5c954d0>

# Menambahkan hasil klasifikasi sebagai overlay
folium.raster_layers.ImageOverlay(
    name='Hasil KMeans',
    image='hasil_kmeans.png',
    bounds=bounds_latlon,
    opacity=1,
    interactive=True,
    cross_origin=False
).add_to(m)

↳ <folium.raster_layers.ImageOverlay at 0x7a88f8197350>
```

```
# Menambahkan kontrol layer
folium.LayerControl().add_to(m)

↳ <folium.map.LayerControl at 0x7a88f5ef5950>

# Simpan Hasil
m.save('peta_interaktif_kmeans.html')
```

## ▼ 4. Evaluasi dan Analisis

Hasil klasifikasi awal dengan Random Forest menunjukkan performa yang cukup baik meskipun tanpa menggunakan Hyperparameter Tuning dengan akurasi keseluruhan 91% dan f1-score rata-rata 0.90. Kelas 2 memiliki hasil terbaik dengan f1-score 0.95, diikuti kelas 0 dengan 0.90. Namun, kelas 1 memiliki performa terendah, terutama pada recall yang hanya 0.75, menunjukkan model sering keliru mendeteksi kelas ini. Confusion matrix juga memperlihatkan sebagian sampel kelas 1 terkласifikasi sebagai kelas lain, kemungkinan karena kemiripan ciri spektral atau jumlah sampel yang lebih sedikit. Secara umum, model sudah baik sebagai baseline, namun masih perlu penyempurnaan untuk meningkatkan hasil pada kelas 1.

Setelah dilakukan hyperparameter tuning, model terbaik diperoleh dengan parameter: max\_depth=10, n\_estimators=200, min\_samples\_split=2, min\_samples\_leaf=1, bootstrap=True, menghasilkan skor validasi silang (Best Score) sebesar 0.88.

Namun, hasil evaluasi pada data uji menunjukkan bahwa performa model tidak banyak berubah dibandingkan sebelum tuning. Akurasi tetap di angka 91%, dengan pola f1-score per kelas yang sama: kelas 2 masih menjadi yang terbaik (0.95), kelas 0 baik (0.90), dan kelas 1 tetapi yang terendah (0.83) terutama pada recall (0.75). Hal ini menunjukkan bahwa model baseline sudah cukup optimal, dan tuning yang dilakukan belum berhasil memperbaiki kelemahan pada kelas 1 secara signifikan. Untuk meningkatkan lagi, mungkin diperlukan strategi lain seperti penyeimbangan data, eksplorasi fitur tambahan, atau teknik ensemble yang lebih kompleks.

---

Untuk Segmentasi citra dengan menggunakan K-Means menggunakan evaluasi yang sedikit berbeda dengan Model Machine Learning sebelumnya dikarenakan metode tersebut merupakan metode Unsupervised Learning. Sehingga metode evaluasi yang digunakan adalah SSE (Sum of Squared Error) atau Elbow Method, dan Silhouette Score. Dari hasil uji SSE dan Silhouette Score, jumlah cluster optimal terlihat pada k=4. Elbow pada SSE cukup jelas di k=4, dan Silhouette Score relatif tinggi pada titik tersebut. Secara visual, hasil segmentasi juga menunjukkan pola spasial yang konsisten dan dapat dibedakan dengan jelas antar cluster. Distribusi ukuran cluster pun merata tanpa dominasi satu cluster yang ekstrem. Hasil ini menunjukkan bahwa model K-Means sudah cukup baik untuk memetakan area homogen di wilayah studi.

