Laporan Proyek Pembelajaran Mesin

Mengklasifikasikan Jenis Kanker Ovarium Dari Hasil Scan Mikroskopis Sampel Biopsi Menggunakan Algoritma CNN

Dibuat Oleh :

Mahes Richmen M.P	**11421002**
Fanny Clara Sinaga	**11421028**
Yennisa Marianti Napitupulu	**11421050**

Sarjana Terapan Teknologi Rekayasa Perangkat Lunak

Fakultas Vokasi

Institut Teknologi Del

DAFTAR ISI

BAB 1 BUSINESS UNDERSTANDING

1. Pengertian Business Understanding

Business Understanding merupakan tahap kritis dalam proses data mining dan analisis data, yang melibatkan pemahaman mendalam tentang tujuan bisnis atau permasalahan yang ingin dipecahkan melalui analisis data. Tahap ini memerlukan pemahaman komprehensif tentang konteks bisnis, kebutuhan pemangku kepentingan, dan tujuan yang ingin dicapai. Dalam konteks analisis data, pemahaman bisnis ini membantu pemilihan metode analisis yang sesuai, menentukan variabel yang relevan, dan merancang strategi untuk menghasilkan wawasan yang dapat digunakan untuk membuat keputusan bisnis yang berbasis informasi. Pemahaman bisnis yang kuat juga membantu mengidentifikasi kendala dan batasan yang mungkin mempengaruhi interpretasi hasil analisis.

1.1 Business Understanding : Mengklasifikasikan Jenis Kanker Ovarium Dari Hasil Scan Mikroskopis Sampel Biopsi

1.1.1 Permasalahan:

1. Ketidakpastian diagnostik:Saat ini,proses diagnostik kanker ovarium dari hasil scan mikroskopis sampel biopsi masih mengandalkan evaluasi manual oleh ahli patologi. Hal ini dapat menyebabkan tingkat ketidakpastian dan variasi dalam diagnosis, yang dapat mempengaruhi pengobatan dan prognosis pasien
2. Waktu dan Sumber Daya: Proses manual memerlukan waktu yang signifikan dan menghabiskan sumber daya manusia yang berharga. Keterbatasan jumlah ahli patologi dan peningkatan jumlah kasus kanker ovarium menimbulkan tantangan dalam penanganan yang efisien

3. Kesalahan Manusia:Keterlibatan manusia dalam interpretasi hasil scan mikroskopis meningkatkan risiko kesalahan diagnosis. Kesalahan ini dapat memiliki dampak serius pada keputusan pengobatan dan kualitas hidup pasien.

1.1.2 Tujuan:

1. Meningkatkan Ketepatan Diagnostik: Mengembangkan sistem klasifikasi otomatis yang dapat mengidentifikasi dan mengklasifikasikan jenis kanker ovarium dari hasil scan mikroskopis sampel biopsi dengan tingkat akurasi yang lebih tinggi daripada metode manual.
2. Efisiensi dan Penghematan Waktu: Mengurangi waktu yang dibutuhkan untuk proses diagnostik dengan menggunakan teknologi otomatisasi. Hal ini akan meningkatkan efisiensi dalam penanganan kasus kanker ovarium, memungkinkan pelayanan yang lebih cepat kepada pasien.
3. Mengurangi Kesalahan Diagnosis: Mengurangi risiko kesalahan manusia dengan mengandalkan teknologi klasifikasi otomatis yang dapat memberikan hasil dengan konsistensi tinggi dan akurasi.

1.2 Pengertian Analysis base table (ABT)

Analysis base table (ABT) adalah tabel datar yang digunakan untuk membangun model analitik dan menilai (memprediksi) perilaku masa depan suatu subjek. Tabel ini mewakili subjek prediksi (misalnya pelanggan) dan menyimpan semua data (variabel) yang menggambarkan subjek ini. Tabel dasar analitik dapat dikembangkan sebagai contoh yang lebih umum yang dapat digunakan untuk memecahkan masalah bisnis secara umum, namun lebih sering dikembangkan untuk memecahkan masalah bisnis yang sangat spesifik.

1.2.1 ABT Untuk Topik:Mengklasifikasikan Jenis Kanker Ovarium Dari Hasil Scan Mikroskopis Sampel Biopsi

1. Descriptive features
    a. image_id:Kode ID unik untuk setiap gambar
    b. image_width:Lebar gambar dalam piksel
    c. image_height:Tinggi gambar dalam piksel
    d. is_tma:untuk menentukan true or false pada gambar

2. Target Feature

   Label:Kelas Sasaran.Salah satu subtipe kanker ovarium ini:CC,EC,HGSC,LGSC,MC,Other.

TABEL ABT:

image_id	label	image_width	mage_height	is_tma

1.2.2 Algoritma yang Digunakan

Kelompok kami memilih algoritma CNN. Convolutional Neural Network (CNN) salah satu jenis neural network yang biasa digunakan pada data image. CNN bisa digunakan untuk mendeteksi dan mengenali objek pada sebuah image.CNN sangat berguna untuk klasifikasi dan pengenalan gambar. CNN sangat efektif dalam tugas pengenalan gambar dan visual, termasuk klasifikasi objek, deteksi objek, dan segmentasi gambar. Kelebihan utamanya adalah kemampuan untuk secara otomatis mempelajari fitur-fitur yang relevan dari data visual tanpa memerlukan ekstraksi fitur manual.

BAB II Data understanding

2. Pengertian Data Understanding

Data understanding merupakan tahap awal dalam proses analisis data yang melibatkan eksplorasi dan pemahaman karakteristik data yang akan digunakan. Tujuan dari tahap ini adalah untuk memahami struktur, sifat, dan potensi data sebelum melakukan analisis lebih lanjut.

2.1 Eksplorasi Data Mengenai Komposisi dan Sebaran Data

Eksplorasi dan deskripsi data merupakan fase awal dalam proses pembelajaran mesin yang bertujuan untuk memahami secara mendalam karakteristik, distribusi, dan struktur data yang akan digunakan untuk mengembangkan model.

2.2 Kode Dan Penjelasan Kode Data Understanding

```
train_df = pd.read_csv("D:PROYEK PM/train.csv")
train_df
```

	image_id	label	image_width	image_height	is_tma
0	4	HGSC	23785	20008	False
1	66	LGSC	48871	48195	False
2	91	HGSC	3388	3388	True
3	281	LGSC	42309	15545	False
4	286	EC	37204	30020	False
...	...	...	...	...	...
533	65022	LGSC	53355	46675	False
534	65094	MC	55042	45080	False
535	65300	HGSC	75860	27503	False
536	65371	HGSC	42551	41800	False
537	65533	HGSC	45190	33980	False

538 rows × 5 columns

```
train_df.nunique()
```

```
image_id        538
label             5
image_width     508
image_height    508
is_tma            2
dtype: int64
```

1. *"train_df = pd.read_csv("D:PROYEK PM/train.csv")*

   train_df

   Kode tersebut digunakan untuk membaca file csv dan menggunakan pandas dan menyimpan datanya dalam DataFrame yang disebut train_df.

2. *train_df.nunique()*

   Kode tersebut digunakan untuk menghitung jumlah nilai unik (distinct) dalam setiap kolom dari DataFrame train_df. Fungsi ini berguna untuk mengetahui berapa banyak nilai yang berbeda dalam setiap kolom data. Hasilnya akan berupa Series pandas yang berisi jumlah nilai unik untuk setiap kolom.

```
train_df.describe()
```

	image_id	image_width	image_height
count	538.000000	538.000000	538.000000
mean	32194.340149	48859.533457	29729.460967
std	18774.950592	20040.989927	10762.899796
min	4.000000	2964.000000	2964.000000
25%	15881.250000	34509.000000	22089.500000
50%	32152.000000	48160.000000	29732.000000
75%	47892.500000	64143.750000	37880.750000
max	65533.000000	105763.000000	50155.000000

```
train_df['label'].value_counts()
```

```
label
HGSC    222
EC      124
CC       99
LGSC     47
MC       46
Name: count, dtype: int64
```

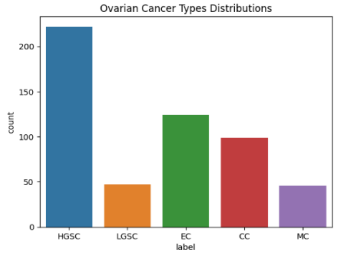
1. *train_df.describe()*

   Kode tersebut digunakan untuk menghasilkan statistik deskriptif (seperti rata-rata, standar deviasi, kuartil) dari kolom numerik dalam DataFrame train_df

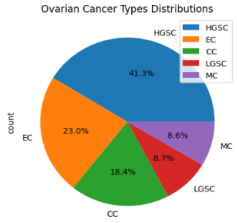
2. *train_df['label'].value_counts()*

   Kode tersebut digunakan untuk menghitung jumlah kemunculan setiap nilai unik dalam kolom 'label' dari DataFrame train_df. Fungsi ini berguna untuk memahami distribusi kategori atau label dalam data.

```
sns.countplot(data=train_df, x="label")
plt.title("Ovarian Cancer Types Distributions")
plt.show()
```




```
train_df['label'].value_counts().plot(kind='pie',autopct='%.1f%%')
plt.title("Ovarian Cancer Types Distributions")
plt.legend()
plt.show()
```



1. *sns.countplot(data=train_df, x="label")*

   plt.title("Ovarian Cancer Types Distributions")

plt.show()

Kode tersebut menggunakan Seaborn untuk membuat diagram batang yang menampilkan distribusi kategori dalam kolom 'label' dari DataFrame train_df

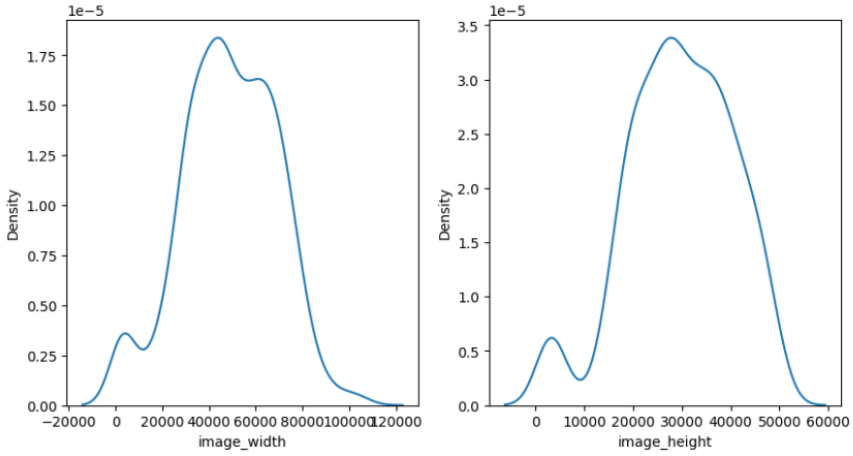
2. *train_df['label'].value_counts().plot(kind='pie',autopct="%.1f%%")*
   plt.title("Ovarian Cancer Types Distributions")
   plt.legend()
   plt.show()

   Kode tersebut membuat diagram lingkaran untuk memvisualisasikan distribusi kategori dalam kolom 'label' dari DataFrame train_df

```
train_df.head()
```

	image_id	label	image_width	image_height	is_tma
0	4	HGSC	23785	20008	False
1	66	LGSC	48871	48195	False
2	91	HGSC	3388	3388	True
3	281	LGSC	42309	15545	False
4	286	EC	37204	30020	False

```
plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
sns.kdeplot(train_df['image_width'])
plt.subplot(1,2,2)
sns.kdeplot(train_df['image_height'])
plt.show()
```



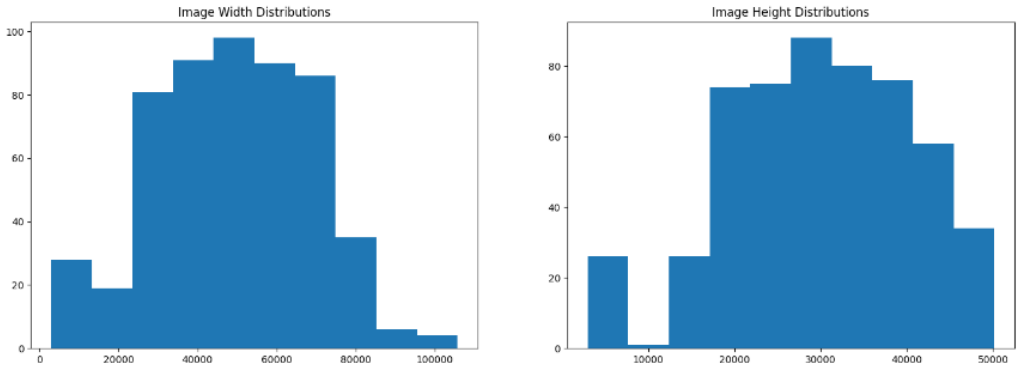
1. *train_df.head()*

   Kode tersebut digunakan untuk menampilkan beberapa baris pertama (biasanya lima baris secara default) dari DataFrame train_df

2. *plt.figure(figsize=(10,5))*
   plt.subplot(1,2,1)
   sns.kdeplot(train_df['image_width'])
   plt.subplot(1,2,2)
   sns.kdeplot(train_df['image_height'])
   plt.show()

Kode ini menggunakan Matplotlib dan Seaborn untuk membuat gambar dengan dua subplot, di mana setiap subplot menampilkan estimasi kepadatan kernel (KDE) dari kolom 'image_width' dan 'image_height' dari DataFrame train_df. Subplot pertama menangani 'image_width', sedangkan subplot kedua menangani 'image_height'

```
plt.figure(figsize=(18,6))
plt.subplot(1,2,1)
plt.hist(x=train_df['image_width'])
plt.title("Image Width Distributions")
plt.subplot(1,2,2)
plt.hist(x=train_df['image_height'])
plt.title("Image Height Distributions")
plt.show()
```




```
train_df['image_width'].describe()
```

```
count       538.000000
mean      48859.533457
std       20040.989927
min        2964.000000
25%       34509.000000
50%       48160.000000
75%       64143.750000
max      105763.000000
Name: image_width, dtype: float64
```

```
train_df['image_height'].describe()
```

```
count      538.000000
mean     29729.460967
std      10762.899796
min       2964.000000
25%      22089.500000
50%      29732.000000
75%      37880.750000
max      50155.000000
Name: image_height, dtype: float64
```

1. *plt.figure(figsize=(18,6))*
   plt.subplot(1,2,1)
   plt.hist(x=train_df['image_width'])
   plt.title("Image Width Distributions")
   plt.subplot(1,2,2)
   plt.hist(x=train_df['image_height'])
   plt.title("Image Height Distributions")
   plt.show()

   Kode ini menggunakan Matplotlib untuk membuat gambar dengan dua subplot yang menampilkan histogram dari kolom 'image_width' dan 'image_height' dari DataFrame train_df. Subplot pertama menangani 'image_width', sedangkan

subplot kedua menangani 'image_height'. Ukuran gambar secara keseluruhan diatur menjadi 18x6 unit.

2. *train_df['image_width'].describe()*

   Kode train_df['image_width'].describe() menghasilkan statistik deskriptif untuk kolom 'image_width' dari DataFrame train_df.

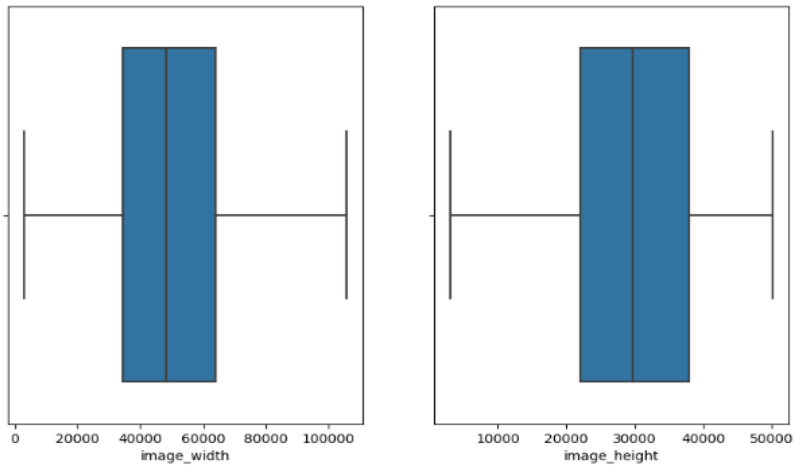
3. *train_df['image_height'].describe()*

   Kode train_df['image_height'].describe() menghasilkan statistik deskriptif untuk kolom 'image_height' dari DataFrame train_df.

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 538 entries, 0 to 537
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   image_id      538 non-null    int64
 1   label         538 non-null    object
 2   image_width   538 non-null    int64
 3   image_height  538 non-null    int64
 4   is_tma        538 non-null    bool
dtypes: bool(1), int64(3), object(1)
memory usage: 17.5+ KB
```

```
plt.figure(figsize=(10,6))
plt.subplot(1,2,1)
sns.boxplot(data=train_df,x="image_width")
plt.subplot(1,2,2)
sns.boxplot(data=train_df,x="image_height")
plt.show()
```



1. *train_df.info()*

   Kode train_df.info() memberikan ringkasan tentang struktur DataFrame train_df, termasuk jumlah entri, tipe data, dan informasi non-null untuk setiap kolom

2. *plt.figure(figsize=(10,6))*

   plt.subplot(1,2,1)

   sns.boxplot(data=train_df,x="image_width")

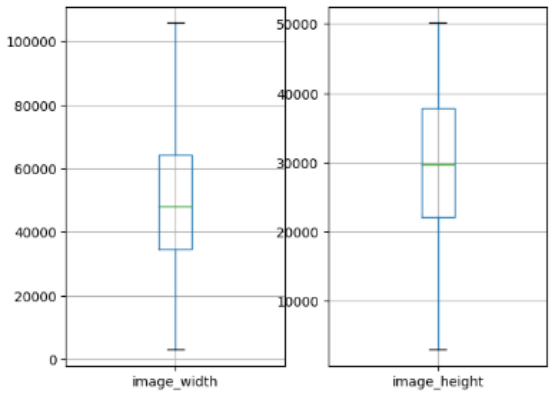
   plt.subplot(1,2,2)

   sns.boxplot(data=train_df,x="image_height")

   plt.show()

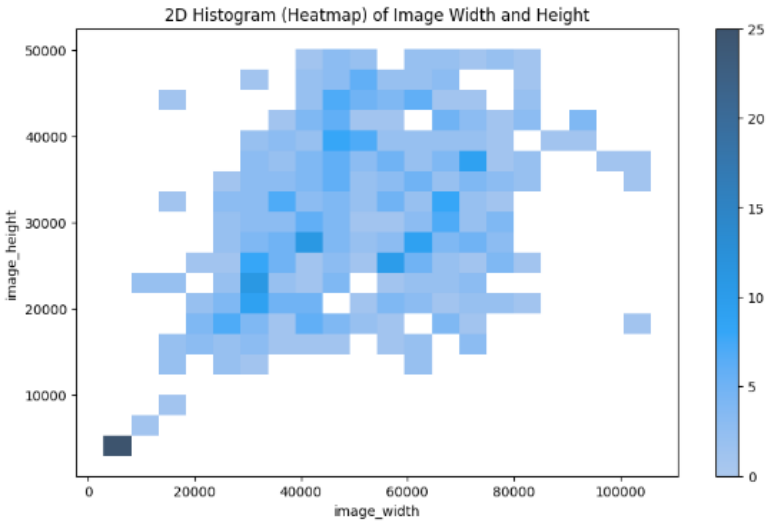
Kode ini menciptakan gambar dengan dua subplot, masing-masing menampilkan diagram kotak dari kolom 'image_width' dan 'image_height' dalam DataFrame train_df. Diagram kotak memberikan informasi tentang distribusi dan statistik deskriptif dari kedua kolom tersebut

```
plt.subplot(1,2,1)
train_df[['image_width']].boxplot()
plt.subplot(1,2,2)
train_df[['image_height']].boxplot()
plt.show()
```




```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
sns.histplot(data=train_df, x="image_width", y="image_height", bins=20, cbar=True)
plt.title("2D Histogram (Heatmap) of Image Width and Height")
plt.show()
```



1. *plt.subplot(1,2,1)*
   train_df[['image_width']].boxplot()
   plt.subplot(1,2,2)
   train_df[['image_height']].boxplot()
   plt.show()

Kode tersebut menciptakan gambar dengan dua subplot, masing-masing menampilkan diagram kotak dari kolom 'image_width' dan 'image_height' dalam DataFrame train_df. Subplot pertama menunjukkan diagram kotak untuk 'image_width', sementara subplot kedua menunjukkan diagram kotak untuk 'image_height'

2. *import seaborn as sns*
   import matplotlib.pyplot as plt
   plt.figure(figsize=(10, 6))
   sns.histplot(data=train_df, x="image_width", y="image_height", bins=20, cbar=True)
   plt.title("2D Histogram (Heatmap) of Image Width and Height")
   plt.show()
   Kode tersebut menggunakan Seaborn dan Matplotlib untuk membuat heatmap (histogram dua dimensi) dari kolom 'image_width' dan 'image_height' dalam DataFrame train_df. Heatmap ini memberikan visualisasi tentang distribusi dan hubungan antara dua kolom tersebut.

BAB III Data preparation

3. Pengertian Data preparation

Data preparation adalah langkah kritis yang memastikan data siap digunakan untuk melatih dan menguji model dengan hasil yang baik, melibatkan proses membersihkan,

mentransformasi, dan mengorganisasi data agar siap digunakan untuk analisis lebih lanjut. Melalui langkah-langkah ini, kita dapat meminimalkan efek negatif dari noise atau ketidaksesuaian data dan meningkatkan keberhasilan.

3.1 Kode Dan Penjelasan Kode Data Preparation

```
]: path_train = "D:/PROYEK PM/train_thumbnails"
   path_test = "D:/PROYEK PM/test_thumbnails"
   train_folder = os.listdir(path_train)
   test_folder = os.listdir(path_test)

   print(len(train_folder))
   print(len(test_folder))

   514
   1

]: train_folder[:5]

]: ['10077_thumbnail.png',
    '10143_thumbnail.png',
    '1020_thumbnail.png',
    '10246_thumbnail.png',
    '10252_thumbnail.png']

]: test_folder

]: ['41_thumbnail.png']
```

1. Definisikan variabel path_train dan path_test yang merujuk ke direktori data yang ingin digunakan.
2. Selanjutnya gunakan *os.listdir()* untuk mendapatkan daftar gambar dari masing-masing direktori.
3. *print(len(train_folder)), print(len(test_folder*)) digunakan untuk mencetak panjang(jumlah elemen) dari variabel yang berisi daftar gambar tersebut.
4. *train_folder[:5]* kode tersebut berguna untuk menampilkan nama-nama dari 5 nilai pertama dari variabel train_folder.
5. *test_folder* kode tersebut digunakan untuk menampilkan semua data yang ada didalam variabel test_folder dari folder test_thumbnails.

```
## Image id >> Tissue Microarray
train_df_tma = train_df[train_df['is_tma']==True]
```

```
train_df_tma
```

	image_id	label	image_width	image_height	is_tma
2	91	HGSC	3388	3388	True
37	4134	MC	2964	2964	True
76	8280	HGSC	2964	2964	True
83	9200	MC	3388	3388	True
112	13568	LGSC	2964	2964	True
149	17637	HGSC	2964	2964	True
176	21020	MC	3388	3388	True
236	29084	LGSC	3388	3388	True
263	31594	EC	3388	3388	True
288	35565	MC	2964	2964	True
299	36302	CC	3388	3388	True
302	36583	LGSC	3388	3388	True
305	36783	MC	2964	2964	True
309	37385	LGSC	3388	3388	True
350	40864	LGSC	2964	2964	True

1. *train_df_tma = train_df[train_df['is_tma'] == True]* kode tersebut membuat variabel baru yaitu train_df_tma yang berisi baris-baris data yang dimana kolom is_tma bernilai True.
2. *train_df_tm* kode berikut berguna untuk menampilkan data dari variable train_df_tma.

```
train_df_no_tma = train_df[train_df['is_tma']==False]
train_df_no_tma
```

	image_id	label	image_width	image_height	is_tma
0	4	HGSC	23785	20008	False
1	66	LGSC	48871	48195	False
3	281	LGSC	42309	15545	False
4	286	EC	37204	30020	False
5	431	HGSC	39991	40943	False

1. *train_df_no_tma = train_df[train_df['is_tma'] == False]* Kode tersebut membuat variabel yaitu variable train_df_no_tma yang berisi baris-baris data yang dimana kolom is_tma bernilai False.
2. *train_df_no_tma* kode berikut berguna untuk menampilkan data dari variabel train_df_no_tma.

```
train_df_no_tma['image_id_path'] = [f"{i}_thumbnail.png" for i in train_df_no_tma['image_id']]
```

```
train_df_no_tma
```

	image_id	label	image_width	image_height	is_tma	image_id_path
0	4	HGSC	23785	20008	False	4_thumbnail.png
1	66	LGSC	48871	48195	False	66_thumbnail.png
3	281	LGSC	42309	15545	False	281_thumbnail.png
4	286	EC	37204	30020	False	286_thumbnail.png
5	431	HGSC	39991	40943	False	431_thumbnail.png
...	...	...	...	...	...	...
533	65022	LGSC	53355	46675	False	65022_thumbnail.png
534	65094	MC	55042	45080	False	65094_thumbnail.png
535	65300	HGSC	75860	27503	False	65300_thumbnail.png
536	65371	HGSC	42551	41800	False	65371_thumbnail.png
537	65533	HGSC	45190	33980	False	65533_thumbnail.png

513 rows × 6 columns

1. *train_df_no_tma['image_id_path] = [f"(i) _thumbnail.png" for i in train_df_no_tma['image_id]]*

   kode berikut berguna untuk membuat kolom baru dengan nama image_id_path yang dimana isinya nanti data dari variable train_df_no_tma sebelumnya yang kolom is_tmanya bernilai False, dan nama tiap datanya akan di set sesuai dengan image_id ditambahkan _thumbnail.png.
2. *train_df_no_tma* kode berikut berguna untuk menampilkan data dari variable train_df_no_tma.

```
train_df_tma['image_id_path'] = [f"{i}.png" for i in train_df_tma['image_id']]
```

```
train_df_tma
```

	image_id	label	image_width	image_height	is_tma	image_id_path
2	91	HGSC	3388	3388	True	91.png
37	4134	MC	2964	2964	True	4134.png
76	8280	HGSC	2964	2964	True	8280.png
83	9200	MC	3388	3388	True	9200.png
112	13568	LGSC	2964	2964	True	13568.png
149	17637	HGSC	2964	2964	True	17637.png
176	21020	MC	3388	3388	True	21020.png
236	29084	LGSC	3388	3388	True	29084.png
263	31594	EC	3388	3388	True	31594.png

1. *train_df_tma['image_id_path] = [f"(i).png" for i in train_df_tma['image_id]]*
   kode berikut berguna untuk membuat kolom baru dengan nama image_id_path yang dimana isinya nanti data dari variable train_df_tma sebelumnya yang kolom is_tmanya bernilai True, dan nama tiap datanya akan di set sesuai dengan image_id.png.
2. *train_df_tma* kode berikut berguna untuk menampilkan data dari variable train_df_tma.

```
image_data = []
image_label = []
path = "D:/PROYEK PM/train_thumbnails"

for img , label in zip(train_df_no_tma['image_id_path'],train_df_no_tma['label']):
    image = Image.open("D:/PROYEK PM/train_thumbnails/"+img)
    image = image.resize((224,224))
    image = image.convert("RGB")
    image = np.array(image)
    image_data.append(image)
    image_label.append(label)
```

1. *image_data = []*
   image_label = []
   mendefinisikan 2 list kosong
2. *path = "D:/PROYEK PM/train_thumbnails"* variabel path akan berisi jalur ke direktori yang berisi file gambar.
3. *for img , label in zip(train_df_no_tma['image_id_path'],train_df_no_tma['label']):* dilakukan looping dari pasangan nilai 2 kolom image_id_path dan label dari train_df_no_tma.
4. *image = Image.open("D:/PROYEK PM/train_thumbnails/"+img)*
   kode tersebut berguna untuk membuka file gambar yang sesuai dengan nilai img dalam loop.
5. *image = image.resize((224,224))* kode tersebut berguna untuk mengubah ukuran gambar menjadi 224x224 piksel.
6. *image = image.convert("RGB")* kode berikut berguna untuk mengonversi gambar ke mode warna RGB.
7. *image = np.array(image)* kode berikut berguna untuk mengonversi gambar menjadi array.
8. *image_data.append(image)* kode berikut berguna untuk menambah array gambar ke dalam list image_data
9. *image_label.append(label*) kode berikut berguna untuk menambahkan label kedalam list image_label.

```
set(image_label)
```

```
{'CC', 'EC', 'HGSC', 'LGSC', 'MC'}
```

```
image_label_1 = []
for i in image_label:
    if i=="CC":
        image_label_1.append(0)
    elif i=="EC":
        image_label_1.append(1)
    elif i=="HGSC":
        image_label_1.append(2)
    elif i=="LGSC":
        image_label_1.append(3)
    elif i=="MC":
        image_label_1.append(4)
```

1. *set(image_label)* kode berikut berguna untuk mendapatkan nilai dari variable image_label yang nilai nya hanya nilai unik jadi akan mengabaikan nilai yang duplikat.
2. *image_label_1 = []* kode berikut akan membuat list kosong yang baru.
3. *for i in image_label:*

   if i=="CC":

   image_label_1.append(0)

   elif i=="EC":

   image_label_1.append(1)

   elif i=="HGSC":

   image_label_1.append(2)

   elif i=="LGSC":

   image_label_1.append(3)

   elif i=="MC":

   image_label_1.append(4)

   kode loop diatas berguna untuk melakukan perulangan dari nilai variable image_label yang dimana terdapat kondisi:

   jika nilai i adalah CC maka akan ditambahkan nilai 0 ke dalam list image_label_1.

   jika nilai i adalah EC maka akan ditambahkan nilai 1 ke dalam list image_label_1.

   jika nilai i adalah HGSC maka akan ditambahkan nilai 2 ke dalam list image_label_1.

   jika nilai i adalah LGSC maka akan ditambahkan nilai 3 ke dalam list image_label_1.

jika nilai i adalah MC maka akan ditambahkan nilai 4 ke dalam list image_label_1.

BAB IV Modeling

4. Pengertian Modeling

Modeling adalah tahap dalam analisis data di mana model atau algoritma pembelajaran mesin digunakan untuk mengambil wawasan atau membuat prediksi berdasarkan data yang telah dipersiapkan sebelumnya. Tahap modeling adalah proses yang iteratif, dan beberapa percobaan mungkin diperlukan untuk mencapai model yang optimal. Selama iterasi, analisis hasil dan evaluasi model membimbing penyesuaian dan perbaikan untuk mencapai performa model yang diinginkan.

4.1 Kode Dan Penjelasan Kode Modeling

```
x_train , x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,shuffle=True)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(410, 224, 224, 3)
(103, 224, 224, 3)
(410,)
(103,)
```

1. *x_train , x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,shuffle=True)*
   print(x_train.shape)
   kode berikut berguna untuk mencetak shape dari array x_train yaitu baris data dari data train.
2. *print(x_test.shape)*
   kode berikut berguna untuk mencetak shape dari array x_test yaitu baris data dari data test.
3. *print(y_train.shape)*
   kode berikut berguna untuk mencetak shape dari array y_train yaitu baris data dari label train.
4. *print(y_test.shape)*
   kode berikut berguna untuk mencetak shape dari array y_test yaitu baris data dari label test.

```
model = Sequential()
model.add(Conv2D(filters=120,kernel_size=(3,3),strides=(1,1),activation='relu',input_shape=(224,224,3)))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(filters=100,kernel_size=(3,3),strides=(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(filters=80,kernel_size=(3,3),strides=(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(filters=64,kernel_size=(3,3),strides=(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(filters=40,kernel_size=(3,3),strides=(1,1),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
model.add(Dense(units=600,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(units=600,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(units=5,activation='softmax'))
model.compile(optimizer="adam",loss="sparse_categorical_crossentropy",
              metrics=["accuracy"])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 222, 222, 120)     3360

 max_pooling2d (MaxPooling2  (None, 111, 111, 120)     0
 D)

 conv2d_1 (Conv2D)           (None, 109, 109, 100)     108100

 max_pooling2d_1 (MaxPoolin  (None, 54, 54, 100)       0
 g2D)

 conv2d_2 (Conv2D)           (None, 52, 52, 80)        72080

 max_pooling2d_2 (MaxPoolin  (None, 26, 26, 80)        0
 g2D)
```

1. *model = Sequential()*

   model.add(Conv2D(filters=120,kernel_size=(3,3),strides=(1,1),activation='relu',input_shape=(224,224,3)))

   model.add(MaxPooling2D(pool_size=(2,2)))

   model.add(Conv2D(filters=100,kernel_size=(3,3),strides=(1,1),activation='relu'))model.add(MaxPooling2D(pool_size=(2,2)))

   model.add(Conv2D(filters=80,kernel_size=(3,3),strides=(1,1),activation='relu'))model.add(MaxPooling2D(pool_size=(2,2)))

   model.add(Conv2D(filters=64,kernel_size=(3,3),strides=(1,1),activation='relu'))model.add(MaxPooling2D(pool_size=(2,2)))

   model.add(Conv2D(filters=40,kernel_size=(3,3),strides=(1,1),activation='relu'))model.add(MaxPooling2D(pool_size=(2,2)))

   model.add(Flatten())

   model.add(Dense(units=600,activation='relu'))

   model.add(Dropout(0.2))

   model.add(Dense(units=600,activation='relu'))

   model.add(Dropout(0.2))

   model.add(Dense(units=5,activation='softmax'))

   model.compile(optimizer="adam",loss="sparse_categorical_crossentropy", metrics=["accuracy"])

   model.summary():

Kode berikut terdiri dari beberapa layer Convolution 2D dan layer MaxPooling2D yang berguna untuk mengekstraksi fitur dan mengurangi dimensi gambar. Setelah serangkaian operasi konvolusi dan pooling, hasilnya diratakan menggunakan layer Flatten. Model juga memasukan layer Dense (fully connected) dengan fungsi aktivitasi relu, serta layer Dropout untuk mencegah overfitting dengan merandomly menonaktifkan beberapa unit neuron selama pelatihan. Layer Dense terakhir menggunakan fungsi aktivasi softmax untuk output klasifikasi multikelas dengan lima kelas yang berbeda. Model ini dikompilasi dengan optimizer Adam, fungsi loss sparse_categorical_crossentropy. model.summary() menunjukkan detail arsitektur dan jumlah parameter yang dapat di-train.

```
model.compile(optimizer ='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

```
epochs = 10
ThisModel = model.fit(x_train, y_train, epochs=epochs,batch_size=64,verbose=1)
```

```
Epoch 1/10
7/7 [==============================] - 46s 6s/step - loss: 0.1112 - accuracy: 0.9634
Epoch 2/10
7/7 [==============================] - 43s 6s/step - loss: 0.1940 - accuracy: 0.9341
Epoch 3/10
7/7 [==============================] - 42s 6s/step - loss: 0.1048 - accuracy: 0.9634
Epoch 4/10
7/7 [==============================] - 46s 7s/step - loss: 0.2014 - accuracy: 0.9439
Epoch 5/10
7/7 [==============================] - 40s 6s/step - loss: 0.1245 - accuracy: 0.9561
Epoch 6/10
7/7 [==============================] - 38s 5s/step - loss: 0.1088 - accuracy: 0.9610
Epoch 7/10
7/7 [==============================] - 43s 6s/step - loss: 0.0981 - accuracy: 0.9537
Epoch 8/10
7/7 [==============================] - 42s 6s/step - loss: 0.2403 - accuracy: 0.9488
Epoch 9/10
7/7 [==============================] - 40s 6s/step - loss: 0.2216 - accuracy: 0.9244
Epoch 10/10
7/7 [==============================] - 40s 6s/step - loss: 0.1510 - accuracy: 0.9366
```

1. *model.compile(optimizer = 'adam',loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])*

   kode tersebut berguna untuk menentukan optimizer yang akan digunakan, menentukan fungsi loss, dan menentukan metriks evaluasi yang akan digunakan selama pelatihan model.

2. *epochs = 10*

   ThisModel = model.fit(x_train, y_train, epochs=epochs,batch_size=64,verbose=1)

   kode berikut akan menyimpan hasil dari proses pelatihan pada variabel ThisModel. Model akan melihat seluruh data latih sebanyak epochs .

BAB V Evaluation

5. Pengertian Evaluation

Evaluasi pada analisis data adalah langkah kunci untuk mengukur sejauh mana model atau hasil analisis dapat diandalkan dan relevan. Evaluasi ini dilakukan untuk memastikan bahwa temuan atau prediksi yang dihasilkan memiliki validitas dan dapat diterima dalam konteks tujuan analisis. Evaluasi pada analisis data adalah langkah kunci untuk mengukur sejauh mana model atau hasil analisis dapat diandalkan dan relevan. Evaluasi ini dilakukan untuk memastikan bahwa temuan atau prediksi yang dihasilkan memiliki validitas dan dapat diterima dalam konteks tujuan analisis.

5.1 Hold-Out Untuk :80:20

```
x_train , x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,shuffle=True)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

1. *x_train,x_test,y_train,y_test =train_test_split(x,y,test_size=0.2,shuffle=True)*

   print(x_train.shape)

   print(x_test.shape)

   print(y_train.shape)

   print(y_test.shape)

   Kode tersebut menggunakan metode hold-out untuk membagi data menjadi dua subset: pelatihan (80%) dan pengujian (20%). Proses ini dilakukan dengan menggunakan fungsi train_test_split dari scikit-learn. Hasilnya adalah empat set data: x_train, x_test, y_train, dan y_test, yang masing-masing merupakan matriks fitur dan variabel target untuk set pelatihan dan pengujian
2. Lalu dalam kode selanjutnya berisi tentang metrikcs performansi

```
ModelLoss, ModelAccuracy = model.evaluate(x_train, y_train)

print('Train Loss is {}'.format(ModelLoss))
print('Train Accuracy is {}'.format(ModelAccuracy ))
```

```
13/13 [==============================] - 15s 977ms/step - loss: 0.9975 - accuracy: 0.6610
Test Loss is 0.9975442886352539
Test Accuracy is 0.6609756350517273
```

```
from sklearn.metrics import precision_score, recall_score, f1_score
import numpy as np

# Assuming you have trained your model and made predictions on the training set
predictions = model.predict(x_train)
y_pred = np.argmax(predictions, axis=1)

# Convert one-hot encoded labels to class labels for the training set
y_true = np.argmax(y_train, axis=1) if len(y_train.shape) > 1 else y_train

# Calculate precision, recall, and F1-score
precision = precision_score(y_true, y_pred, average='weighted')
recall = recall_score(y_true, y_pred, average='weighted')
f1 = f1_score(y_true, y_pred, average='weighted')

# Print the results
print('Precision: {:.4f}'.format(precision))
print('Recall: {:.4f}'.format(recall))
print('F1-Score: {:.4f}'.format(f1))
```

```
13/13 [==============================] - 10s 795ms/step
Precision: 0.5182
Recall: 0.4976
F1-Score: 0.4818
```

5.2 Hold-Out Untuk :90:10

1. *x_train,x_test,y_train,y_test =train_test_split(x,y,test_size=0.1,shuffle=True)*

   print(x_train.shape)

   print(x_test.shape)

   print(y_train.shape)

   print(y_test.shape)

   Kode tersebut menggunakan metode hold-out untuk membagi data menjadi dua subset: pelatihan (90%) dan pengujian (10%). Proses ini dilakukan dengan menggunakan fungsi train_test_split dari scikit-learn. Hasilnya adalah empat set data: x_train, x_test, y_train, dan y_test, yang masing-masing merupakan matriks fitur dan variabel target untuk set pelatihan dan pengujian

2. Lalu dalam kode selanjutnya berisi tentang metrikcs performansi

```
x_train , x_test, y_train, y_test = train_test_split(x,y,test_size=0.1,shuffle=True)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
ModelLoss, ModelAccuracy = model.evaluate(x_train, y_train)

print('Train Loss is {}'.format(ModelLoss))
print('Train Accuracy is {}'.format(ModelAccuracy ))
```

```
15/15 [==============================] - 16s 830ms/step - loss: 0.9627 - accuracy: 0.6182
Test Loss is 0.9627009630203247
Test Accuracy is 0.6182212829589844
```

```python
from sklearn.metrics import precision_score, recall_score, f1_score
import numpy as np

# Assuming you have trained your model and made predictions on the training set
predictions = model.predict(x_train)
y_pred = np.argmax(predictions, axis=1)

# Convert one-hot encoded labels to class labels for the training set
y_true = np.argmax(y_train, axis=1) if len(y_train.shape) > 1 else y_train

# Calculate precision, recall, and F1-score
precision = precision_score(y_true, y_pred, average='weighted')
recall = recall_score(y_true, y_pred, average='weighted')
f1 = f1_score(y_true, y_pred, average='weighted')

# Print the results
print('Precision: {:.4f}'.format(precision))
print('Recall: {:.4f}'.format(recall))
print('F1-Score: {:.4f}'.format(f1))
```

```
15/15 [==============================] - 13s 818ms/step
Precision: 0.6856
Recall: 0.6182
F1-Score: 0.5850
```

BAB VI Kesimpulan

6. Kesimpulan

Dalam proyek ini,tujuannya adalah memahami dan merinci permasalahan dari sebuah studi kasus.Dengan menetapkan tujuan yang jelas,yaitu pemecahan masalah tertentu maka dilakukan pendefinisian kerangka kerja untuk langkah-langkahnya.Mulai dari data understanding dimana melakukan eksplorasi data,lalu data preparation untuk penanganan data,kemudian modeling dimana merupakan pembagian data training dan testing dalam proyek ini digunakan holdout untuk evaluation experiment,dan terakhir adalah evaluation untuk mengetahui sejauh mana model memenuhi tujuan proyek.Kesimpulan ini mencerminkan pemahaman yang mendalam tentang data, kesuksesan pemodelan, dan evaluasi performa yang memberikan landasan untuk rekomendasi lebih lanjut atau implementasi praktis. Proyek ini menggambarkan pendekatan yang terstruktur dan informatif dalam rangka memecahkan tantangan melalui analisis data dan pemodelan.Dalam hal ini hold-out:80 untuk training dan 20 untuk testing lebih cocok digunakan dikarenakan *accuracy* yang lebih tinggi.