BAB 2

Feature Extraction

1.1. Pengertian Fitur dan Ekstraksi fitur

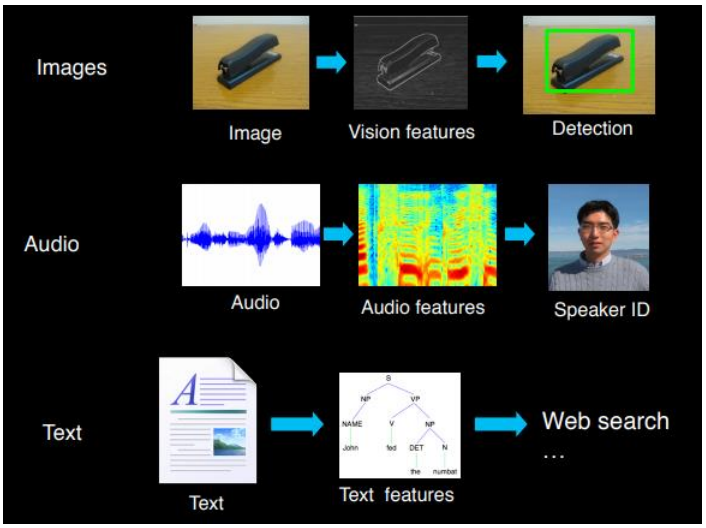
Dalam machine learning untuk melakukan pengolahan terhadap dataset, membuat model pelatatihan dari dataset diperlukan feature. Feature merupakan attribut, karakteristik, ciri, atau sesuatu yang special dari sebuah object. Ketika dataset digunakan untuk membuat model pelatihan dalam Machine learning, bisa saja tidak semua data digunakan. Dipilih beberapa data tertentu yang dapat mewakili dataset tersebut. Data yang dipilih untuk melatih model machine learning tersebut disebut Feature.

Contoh data untuk prediksi harga rumah, terdatapat data dengan karakteristik berikut ini :

- Luas tanah
- Luas bangunan
- Jumlah Kamar Tidur
- Ukuran Torrent Air

Berdasarkan contoh data diatas jika ingin melakukan prediksi harga rumah maka feature yang dapat digunakan adalah adalah luas tanah, luas bangunan, dan jumlah kamar tidur. Akan tetapi tentunya feature ukuran torrent air tidak digunakan karena torrent air tidak akan mempengaruhi harga rumah.

Contoh :



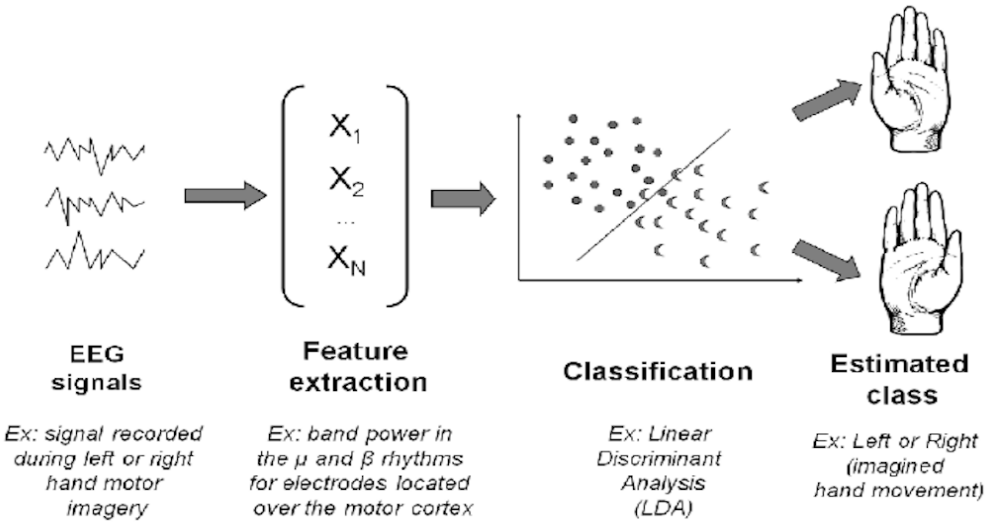
Gambar 1 . Fitur untuk Machine Learning

Sumber : Stanford

Ekstraksi Fitur

Ekstraksi Fitur merupakan sebuah proses pengurangan dimensi dimana data awal yang berupa sekumpulan data mentah kemudian direduksi menjadi kelompok yang lebih mudah dikelola untuk diproses oleh computer.

Contoh ekstraksi fitur



Gambar 2 . Contoh Ekstraksi Fitur dari sinyal EEG

Pada gambar 2 terlihat bahwa data awal berupa sinya EEG. Sinyal EEG (electroencephalogram) merupakan salah satu bentuk informasi dari otak manusia, sinyal ini memiliki karakteristik sebagai gelombang elektromagnetik. Namun pemrosesan sinyal EEG tidak mudah kadang terdapat noise saat perekaman data sehingga dibutuhkan ekstraksi fitur yang dapat merepresentasikan sinyal EGG. Setelah fitur didapat kemudian dilakukan klasifikasi yang kemudian menghasilkan prediksi pergerakan tangan kanan atau kiri.

1.2. Feature Scaling

Feature scaling adalah Teknik untuk menstandarisasi fitur yang ada dalam data sehingga data tersebut berada dalam rentang tetap. Hal ini dilakukan karena terdapat kemungkinan bahwa data diambil dari domain yang berbeda untuk setiap variabel. Variabel masukan mungkin memiliki satuan yang berbeda (misalnya meter, kaki, kilometer, dan jam) sehingga menyebabkan variable-variabel tersebut memiliki skala yang berbeda.

Perbedaan skala antar variabel input dapat meningkatkan kesulitan dalam pemodelan machine learning. Contohnya adalah nilai masukan yang besar (mis., ratusan atau ribuan unit) dapat menghasilkan model yang cenderung terhadap nilai bobot yang besar. Model

dengan nilai bobot yang besar sering kali tidak stabil, dan menganggap nilai yang lebih kecil sebagai nilai yang lebih rendah. Sehingga memungkinkan model tersebut memiliki performa yang buruk sehingga mengakibatkan error yang lebih tinggi.

Contoh: Jika suatu algoritma tidak menggunakan metode Feature SCALING, algoritma dapat menganggap nilai 3000meter lebih besar dari 5 km tentu saja hal tersebut tidak benar dan dapat mengakibatkan algoritma memberikan prediksi yang salah. Sehingga Feature Scaling digunakan untuk membuat nilai ke besaran yang sama dan dengan demikian diaharapkan dapat mengatasi performa yang buruk dari algoritma akibat skala variable yang berbeda.

Perbedaan skala untuk variabel input ini tidak memengaruhi semua algoritma Machine Learning. Contoh algoritma yang performanya terpengaruh perbedaan skala adalah algoritma-algoritma yang melakukan penjumlahan bobot variabel input seperti linear regression, logistic regression, and artificial neural networks (deep learning). Begitu juga algoritma yang menggunakan jarak antar data untuk melakukan prediksi seperti k-nearest neighbours(KNN) and support vector machines(SVM). Ada juga algoritma yang tidak terpengaruh oleh skala variabel input yaitu decision tree dan random forest. Teknik Feature SCALING yang sering digunakan adalah **Normalisasi dan Standardisasi**

A. Normalisasi

   Normalisasi adalah salah satu Teknik feature scaling dimana pada Teknik ini dilakukan penskalaan ulang data dari rentang awal yang kemudian menghasilkan nilai yang berada dalam rentang baru, yaitu antara 0 dan 1.

   Rumus Normalisasi :

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

   Dimana x' adalah skala data dalam rentang baru, X adalah nilai data yang akan dinormalisasi, Xmax adalah nilai maksimal dari variable, Xmin adalah nilai minimal dari variable.

   Misalnya, untuk sekumpulan data, diketahui nilai minimum adalah -10 dan nilai maksimunya adalag 30. Jika data yang akan dinormalisasi adalah 18,8. Maka perhitungan normalisasinya adalah sebagai berikut:

   $X' = (X - Xmin) / (Xmax - Xmin)$

X' = (18.8 – (-10)) / (30 – (-10))

X' = 28.8 / 40

X' = 0.72

Normalisasi menggunakan library scikit-learn MinMaxScaler.

Contoh kode program :

```
# example of a normalization
from numpy import asarray
from sklearn.preprocessing import MinMaxScaler
# define data
data = asarray([[100, 0.001],
                [8, 0.05],
                [50, 0.005],
                [88, 0.07],
                [4, 0.1]])
print(data)
# define min max scaler
scaler = MinMaxScaler()
# transform data
scaled = scaler.fit_transform(data)
print(scaled)
```

```
[[1.0e+02 1.0e-03]
 [8.0e+00 5.0e-02]
 [5.0e+01 5.0e-03]
 [8.8e+01 7.0e-02]
 [4.0e+00 1.0e-01]]
[[1.         0.        ]
 [0.04166667 0.49494949]
 [0.47916667 0.04040404]
 [0.875      0.6969697 ]
 [0.         1.        ]]
```

B. Standardisasi

- Learning algorithms bekerja lebih baik ketika dilatih pada data standar, Standarisasi melibatkan penskalaan ulang distribusi nilai, sehingga didapatkan nilai mean dari data yang diamati adalah 0 dan nilai standar deviasinya adalah 1. Seperti normalisasi, standardisasi dapat berguna dan diperlukan dalam beberapa algoritma machine learning saat data yang digunakan memiliki rentang skala yang jauh berbeda. Berikut adalah rumus "centre scaling " yang dapat digunakan untuk standardisasi data :

$$X' = \frac{X - \mu}{\sigma}$$

- Dimana μ adalah rata-rata dari nilai feature, dan σ adalah nilai standar deviasi dari nilai feature

$Mean(\mu) = \frac{\sum x}{n}$

$standard\_deviation(\sigma) = \sqrt{\frac{\sum(x-\mu)^2}{n}}$

Dimana n adalah banyaknya data

Contoh :

Jika diketahui nilai rata-ratadari sekumpulan data adalah 10,0. Kemudian standar deviasinya adalah 5,0. Jiak sebuah data x memiliki nilai 20,7. Maka dengan menggunakan nilai-nilai ini, dapat dihitung standardisasinya

X′ = (x - mean) / standar deviasi

X′ = (20,7 - 10) / 5

X′ = (10,7) / 5

X′ = 2,14

Implementasi standardisasi menggunakan **StandardScaler**

```
from numpy import asarray
from sklearn.preprocessing import StandardScaler
# define data
data = asarray([[100, 0.001],
                [8, 0.05],
                [50, 0.005],
                [88, 0.07],
                [4, 0.1]])
print(data)
# define standard scaler
scaler = StandardScaler()
# transform data
scaled = scaler.fit_transform(data)
print(scaled)
```

```
[[1.0e+02 1.0e-03]
 [8.0e+00 5.0e-02]
 [5.0e+01 5.0e-03]
 [8.8e+01 7.0e-02]
 [4.0e+00 1.0e-01]]
[[ 1.26398112 -1.16389967]
 [-1.06174414  0.12639634]
 [ 0.         -1.05856939]
 [ 0.96062565  0.65304778]
 [-1.16286263  1.44302493]]
```

1.3. Ekstraksi Fitur dari Data Kategorik

Dalam machine learning terdapat dua jenis data yang sering digunakan yaiyu tipe Data Kategorik dan tipe Data Numerik. Pemahaman terhadap kedua tipe data ini sangatlah penting karena akan berdampak kepada model machine learning yang akan digunakan. Data numerik melibatkan fitur yang hanya terdiri dari angka, seperti bilangan bulat atau bilangan decimal. Tipe data kategorik adalah atribut yang diperlakukan sebagai simbol berbeda atau hanya nama. Data Kategorik digunakan untuk data yang tidak dapat dihitung secara kuantitatif sehingga tidak dapat menerima operasi matematika seperti penjumlahan atau perkalian. Namun demikian, nilai-nilainya dapat dibedakan antara satu dengan lainnya. Berikut merupakan contoh data kategorik :

- Golongan darah pada manusia : A, B, AB, dan O.
- Nilai Huruf yang digunakan pada Politeknik Negeri Malang: A, B+, B, C+, C, D, dan E.
- Posisi podium Balapan MotoGP: pertama, kedua, dan ketiga.

Variabel numerik dapat diubah menjadi variabel ordinal dengan membagi rentang variabel numerik menjadi beberapa bin dan menetapkan nilai ke setiap bin. Misalnya, variabel numerik antara 1 sampai 20 dapat dibagi menjadi variabel ordinal dengan 4 label dengan hubungan ordinal: 1-5, 6-10, 11-15, 16-20. Proses Ini disebut diskritisasi.

Oleh karena itu, dapat didefinisikan bahwa:

- Variabel Nominal (Kategorikal). Variabel terdiri dari sekumpulan nilai diskrit terbatas tanpa hubungan antar nilai. Contohnya adalah data nama kota seperti Jakarta, Bandung, Bali
- Variabel Ordinal. Variabel terdiri dari sekumpulan nilai diskrit yang terbatas dengan urutan peringkat antar nilai. Contohnya variable ordinal adalah Ketika terdapat urutan Low, Medium, High.

Beberapa implementasi algoritma Machine learning mengharuskan semua data harus menjadi numerik. Dalam artian bahwa data kategorik harus diubah ke dalam bentuk numerik. Ada tiga pendekatan umum yang dapat digunakan untuk melakukan konversi variabel ordinal dan variable kategorik menjadi nilai numerik. Yaitu :

- Ordinal Encoding
- One-Hot Encoding
- Dummy Variable Encoding

1. ORDINAL ENCODING

DALAM ordinal encoding, setiap kategori yang unik diberi nilai integer. Misalnya, "merah" adalah 1, "hijau" adalah 2, dan "biru" adalah 3. Biasaya nilai integer yang digunakan berawal dai nilai 0. Ordinal encoding lebih cocok untuk data bertipe Variabel nominal dimana tidak terdapat hubungan atau urutan antar variable.

Contoh:

terdapat data dengan tipe kategorik yaitu “State Polytechnics of Malang”, “Electronic State Polytechnics of Surabaya” dan “State Polytechnics of Jakarta”. Data tersebut akan dirubah menjadi data dalam bentuk numerik. Langkah pertama yang dilakukan adalah

melakukan pengurutan data berdasarkan huruf abjad. Setelah diurutkan maka urutan pertama akan diberi nilai 0, dan seterusnya. Sehingga didapatkan Electronic State Polytechnics of Surabaya =0, State Polytechnics of Jakarta =1, dan State Polytechnics of Malang =2.

scikit-learn Python telah memiliki library untuk melakukan transformasi data kategorik ke data numerical yaitu OrdinalEncoder. Contoh kode program :

```
from sklearn.preprocessing import OrdinalEncoder
ordinal_encoder = OrdinalEncoder()

oe = [
    ['State Polytechnics of Malang'],
    ['Electronic State Polytechnics of Surabaya'],
    ['State Polytechnics of Jakarta']
]

print(ordinal_encoder.fit_transform(oe))
```

Output dari kode program :

```
[[2.]
 [0.]
 [1.]]
```

2. ONE – HOT ENCODING

One-hote encoding variabel direpresentasikan menggunakan satu fitur biner untuk setiap nilai yang mungkin. Untuk data kategorikal (Variabel Nominal) dimana tidak ada hubungan urutan peringkat antar nilai penggunakan ORDINAL ENCODING tidak memberikan performa yang bagus pada model machine learning. Memaksakan hubungan urutan (ordinal) melalui ordinal encoding memungkinkan model untuk berasumsi bahwa terdapat urutan antar kategori sehingga mengakibatkan kinerja yang buruk atau hasil yang tidak diharapkan. Dalam hal ini, **ONE – HOT ENCODING** dapat diterapkan terhadap data yang memiliki tipe Ordinal. Tabel 1.1 menunjukkan contoh one-hot encoding.

State Polytechnics of Malang	0	0	1
Electronic State Polytechnics of Surabaya	1	0	0
State Polytechnics of Jakarta	0	1	0

Table 1. One Hot Encoding

Langkah pertama yang dilakukan adalah melakukan pengurutan data berdasarkan huruf abjad. Setelah diurutkan selanjutnya nilai biner akan ditambahkan kepada setiap kategori. One Hot Encoding akan menambah kolom fitur sesuai dengan nama politeknik yang ada di data. Nilai 1 menunjukan bahwa pada baris tersebut terdapat data politeknik tersebut sedangkan nilai 0 menunjukan sebaliknya. Artinya bahwa Electronic State Polytechnics of Surabaya akan direpresentasikan dengan [1, 0, 0] dengan "1" in untuk nilai biner pertama, kemudian State Polytechnics of Jakarta direpresentasikan dengan [0, 1, 0], dan selanjutnya State Polytechnics of Malang direpresentasikan dengan [0, 0, 1]. Contoh lain dapat dilihat pada tabel 2

	Bandung	Jakarta	Malang	Surabaya	Yogyakarta
0	0	1	0	0	0
1	1	0	0	0	0
2	0	0	0	0	1
3	0	0	0	1	0
4	0	0	1	0	0

Tabel 2. One Hot Encoding pada data kota

scikit-learn Python telah memiliki library untuk melakukan transformasi one hot encoding yaitu OneHotEncoder class. Contoh kode program :

```
from sklearn.feature_extraction import DictVectorizer
onehot_encoder = DictVectorizer()

oe = [
    {'name': 'State Polytechnics of Malang'},
    {'name': 'Electronic State Polytechnics of Surabaya'},
    {'name': 'State Polytechnics of Jakarta'}
]

print(onehot_encoder.fit_transform(oe).toarray())
```

Seperti gambar dibawah, output terdiri dari tiga variabel biner

```
[[0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]]
```

3. DUMMY VARIABLE ENCODING

one-hot encoding membuat satu variabel biner untuk setiap kategori. Terdapat redudansi dalam one-hot encodingi. Contoh jika [1, 0, 0] mewakili "Electronic State

Polytechnics of Surabaya" dan [0, 1, 0] mewakili "State Polytechnics of Jakarta". Maka untuk merepresentasikan "State Polytechnics of Malang" tidak diperlukan nilai biner lainnya, sebagai gantinya bisa digunakan nilai 0 untuk "ectronic State Polytechnics of Surabaya" atau " State Polytechnics of Jakarta"saja, misal. [0, 0]. Cara ini disebut variabel dummy encoding

Tabel 3 merupakan contoh dummy variable encoding.

State Polytechnics of Malang	0	1
Electronic State Polytechnics of Surabaya	0	0
State Polytechnics of Jakarta	1	0

Table 3 Dummy Variable Encoding

Dummy variable encoding dapat diimplementasikan menggunakan scikit-learn dapat dengan menggunakan OneHotEncoder class. Gunakan argument "drop" untuk menunjukkan kategori mana yang akan datang yang diberi nol semua, kategori yang diberi nilai 0 ini disebut baseline. Argumen "drop" dapat diberi nilai "first" yang artinya kategori pertama yang akan diberi nilai 0. Pada contoh kategori politeknik maka "Electronic State Polytechnics of Surabaya" akan diberi nilai 0 dan akan menjadi baseline

Contoh kode program :

```
from sklearn.preprocessing import OneHotEncoder
dummy_encoding = OneHotEncoder(drop='first')

de = [
    ['State Polytechnics of Malang'],
    ['Electronic State Polytechnics of Surabaya'],
    ['State Polytechnics of Jakarta']
]

print(dummy_encoding.fit_transform(de).toarray())
```

Seperti gambar dibawah, output terdiri dari dua buah variable biner

```
[[0. 1.]
 [0. 0.]
 [1. 0.]]
```

1.4. Ekstraksi Fitur pada data Text

Data text seringkali membutuhkan preprocessing sebelum data tersebut dilatih dengan model pembelajaran machine learning. Text preprocessing bertujuan untuk membuat dokumen

masukan lebih konsisten dan mempermudah representasi teks. Text processing secara tradisional dapat dilihat ini berisi tiga proses utama yaitu tokenizing, stopword removal, dan stemming.

- Tahap tokenizing adalah tahap pemotongan string berdasarkan tiap kata yang menyusunnya.
- Stopword removal mengeliminasi kata – kata yang tidak penting berdasarkan daftar stopword. Contoh stopword adalah "yang", "dan", "di", "dari" dan seterusnya.
  Contoh proses stopword removal

Sebelum	Sesudah
|ppdb| |dengan| |sistem| |zonasi| |di| |wilayah| |dki| |jakarta| |ukurannya| |bukan| |lagi| |jarak| |dari| |rumah| |ke| |sekolah|	|ppdb| |sistem| |zonasi| |wilayah| |dki| |jakarta| |ukurannya| |jarak| |rumah| |sekolah|
|mohon| |di| |jawab| |bapak| |apakah| |sistem| |zonasi| |ppdb| |sma| |jakarta| |akan| |memprioritaskan| |peserta| |yang| |terdekat| |tempat| |tinggalnya| |dengan| |sekolah|	|mohon| |sistem| |zonasi| |ppdb| |sma| |jakarta| |akan| |memprioritaskan| |peserta| |terdekat| |tinggalnya| |sekolah|
|hai| |admin| |mengenai| |sistem| |zonasi| |ppdb| |jalur| |umum| |yang| |menjadi| |pertimbangan| |itu| |radius| |kamu| |atau| |harus| |satu| |regional| |iya| |contohnya| |sekolah| |yang| |bertempat| |di| |jakarta| |pusat| |lebih| |memprioritaskan| |calon| |siswa| |dengan| |kk| |di| |jakarta| |pusat| |juga| |kah| |terima| |kasih|	|admin| |sistem| |zonasi| |ppdb| |jalur| |pertimbangan| |radius| |regional| |contohnya| |sekolah| |bertempat| |jakarta| |pusat| |memprioritaskan| |calon| |siswa| |kk| |jakarta| |pusat| |terima| |kasih|

Table 4 Contoh Stopword Removal

- Tahap yang terakhir adalah stemming dimana tahap ini dilakukan untuk mencari root kata. Setiap kata yang memiliki imbuhan seperti imbuhan awalan dan akhiran maka akan diambil kata dasarnya

Text Representation Cara yang paling umum untuk memodelkan dokumen adalah mengubah setiap kata (term) menjadi vektor numerik. Representasi ini disebut "Bag Of Words" (BOW) atau "Vector Space Model" (VSM). Dalam VSM setiap kata yang terdapat didalam dokumen merupakan representasi dari fitur yang berbeda tanpa dipertimbangkan hubungan semantik antar kata yang ada di dalam dokumen. Selanjutnya setiap kata akan direpresentasikan dengan bobot. Metode pembobotan kata yang paling populer adalah Term frequency-inverse document frequency (tf idf)

$$tfidf(t_k) = tf * \log\frac{N}{df(t_k)}$$

Dimana tfidf(tk) merupakan bobot term w didalam dokumen, tf merupakan frekuensi kemunculan term tk didalam sebuah dokumen, dan df adalah jumlah dokumen yang memiliki kata tk.

Contoh perhitungan tfidf

Document 1 :blackberry gemini warna merah.
Document 2 :nokia lumia fitur kamera.
Document 3 :blackberry bold garansi tam.
Document 4 :samsung galaxy tipe layar sentuh.
Document 5 :nokia warna putih.

Menghitung nilai Term frekuensi

Document 1 :blackberry gemini warna merah.
1 1 1 1
Document 2 :nokia lumia fitur kamera.
1 1 1 1
Document 3 :blackberry bold garansi tam.
1 1 1 1
Document 4 :samsung galaxy tipe layar sentuh.
1 1 1 1 1
Document 5 :nokia warna putih.
1 1 1

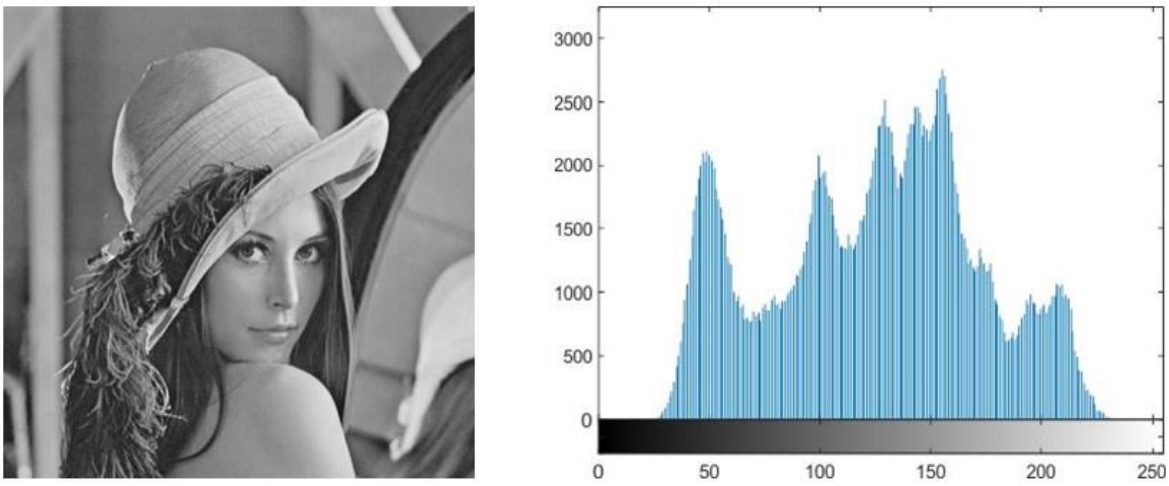
Hasil akhir tfidf

	TF								W						
Token	KK	D1	D2	D3	D4	D5	Df	D/df	IDF	KK	D1	D2	D3	D4	D5
blackberry	1	1	0	1	0	0	2	2.5	0.397	0.397	0.397	0	0.397	0	0
gemini	0	1	0	0	0	0	1	5	0.698	0	0.698	0	0	0	0
warna	1	1	0	0	0	1	2	2.5	0.397	0.397	0.397	0	0	0	0.397
merah	0	1	0	0	0	0	1	5	0.698	0	0.698	0	0	0	0
nokia	0	0	1	0	0	1	2	2.5	0.397	0	0	0.397	0	0	0.397
lumia	0	0	1	0	0	0	1	5	0.698	0	0	0.698	0	0	0
fitur	0	0	1	0	0	0	1	5	0.698	0	0	0.698	0	0	0
kamera	0	0	1	0	0	0	1	5	0.698	0	0	0.698	0	0	0
bold	0	0	0	1	0	0	1	5	0.698	0	0	0	0.698	0	0
garansi	1	0	0	1	0	0	1	5	0.698	0.698	0	0	0.698	0	0
tam	1	0	0	1	0	0	1	5	0.698	0.698	0	0	0.698	0	0
samsung	0	0	0	0	1	0	1	5	0.698	0	0	0	0	0.698	0
galaxy	0	0	0	0	1	0	1	5	0.698	0	0	0	0	0.698	0
tipe	1	0	0	0	1	0	1	5	0.698	0.698	0	0	0	0.698	0
layar	0	0	0	0	1	0	1	5	0.698	0	0	0	0	0.698	0
sentuh	0	0	0	0	1	0	1	5	0.698	0	0	0	0	0.698	0
putih	1	0	0	0	0	1	1	5	0.698	0.698	0	0	0	0	0.698

Table 4 Contoh perhitungan TF-IDF

1.5. Ekstraksi Fitur pada Citra

Salah satu cara melalukan ekstaksi fitur citra adalah dengan menggunakan histrogram. Histogram menunjukkan distribusi piksel berdasarkan intensitas graylevel (derajat keabuan) yang dimiliki oleh tiap-tiap piksel. Pada metode ekstraksi ciri histogram, bin merupakan banyaknya batang warna yang akan terbentuk, atau menunjukkan jumlah pembagian rentang warna pada histogram. Jumlah titik ekstraksi ciri yang dihasilkan oleh suatu histogram adalah sama dengan jumlah bin yang digunakan pada histogram tersebut.



Gambar 3. Contoh Histogram pada citra Grayscale

Contoh perhitungan histogram citra. Diketahui citra pada gambar dengan ukuran 10x10 dengan nilai pixel antara 0-7. Langkah pertama yang dilakukan adalah menghitung frekuensi kemunculan untuk setiap pixel seperti pada tabel 5 dimana x adalah nilai pixel dan y adalah jumlah kemunculan pixel (x). Selanjutnya buat diagram batang dimana kurva x adalah pixel sedangkan y adalah frekuensi kemunculan seperti pada gambar 5.

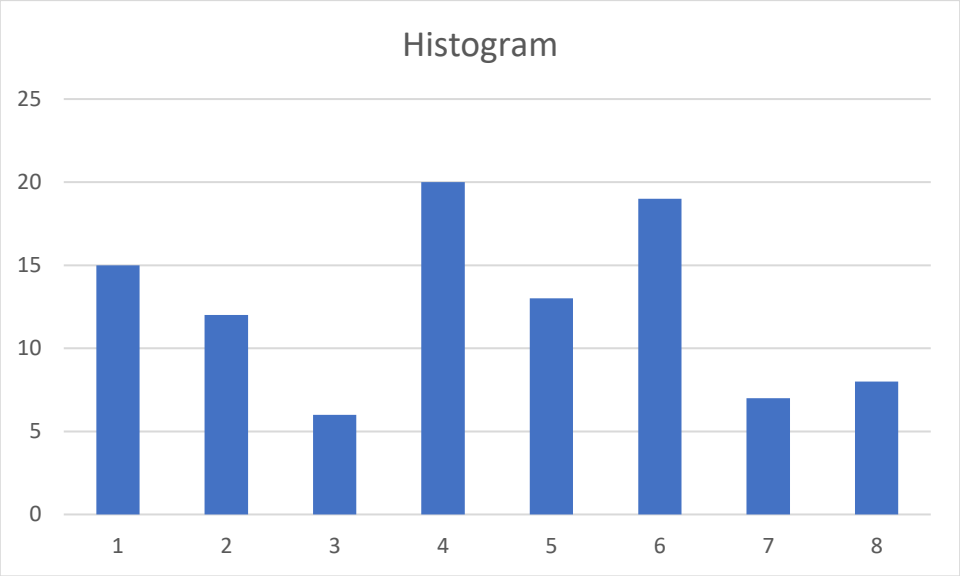
1	1	1	3	1	4	4	4	1	0
3	5	3	5	5	5	5	7	7	0
0	0	0	2	2	6	6	6	6	6
5	5	4	4	4	4	4	4	7	3
2	2	0	0	0	0	1	1	1	1
7	5	5	5	7	7	7	6	3	3
3	3	3	3	3	3	3	3	7	5
5	5	5	5	5	5	5	5	2	3
0	0	0	0	0	0	4	4	4	4
3	3	3	3	3	1	1	1	6	2

Citra

Gambar 4. Citra dengan ukuran 10x10

x	0	1	2	3	4	5
y	15	12	6	20	13	19

Table 5. Frekuensi kemunculan setiap pixel



Gambar 5. Hasil Histogram