JOBSHEET 1
Feature Extraction

1. Tujuan

- Mahasiswa memahami tentang konsep ekstraksi fitur
- Mahasiswa mampu melakukan perhitungan manual ekstraksi fitur
- Mahasiswa mampu mengimplementasikan metode ekstraksi fitur

2. Materi

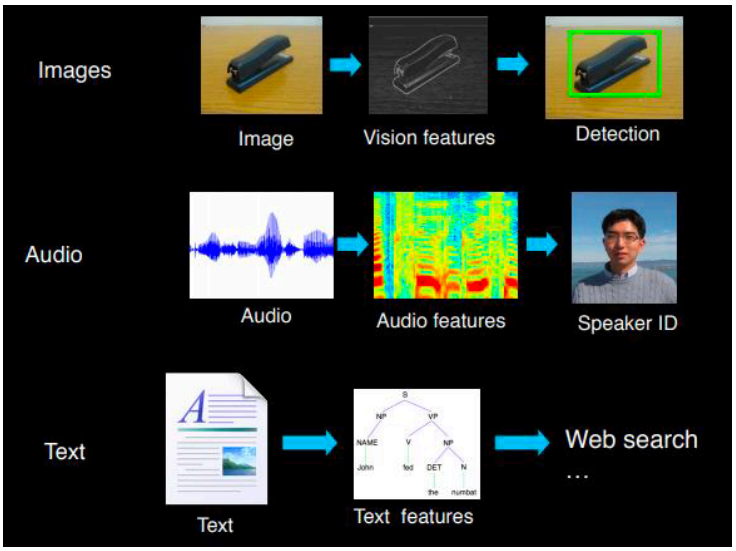
2.1. Feature

Dalam machine learning untuk melakukan pengolahan terhadap dataset, membuat model pelatihan 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, terdapat data dengan karakteristik berikut ini:

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

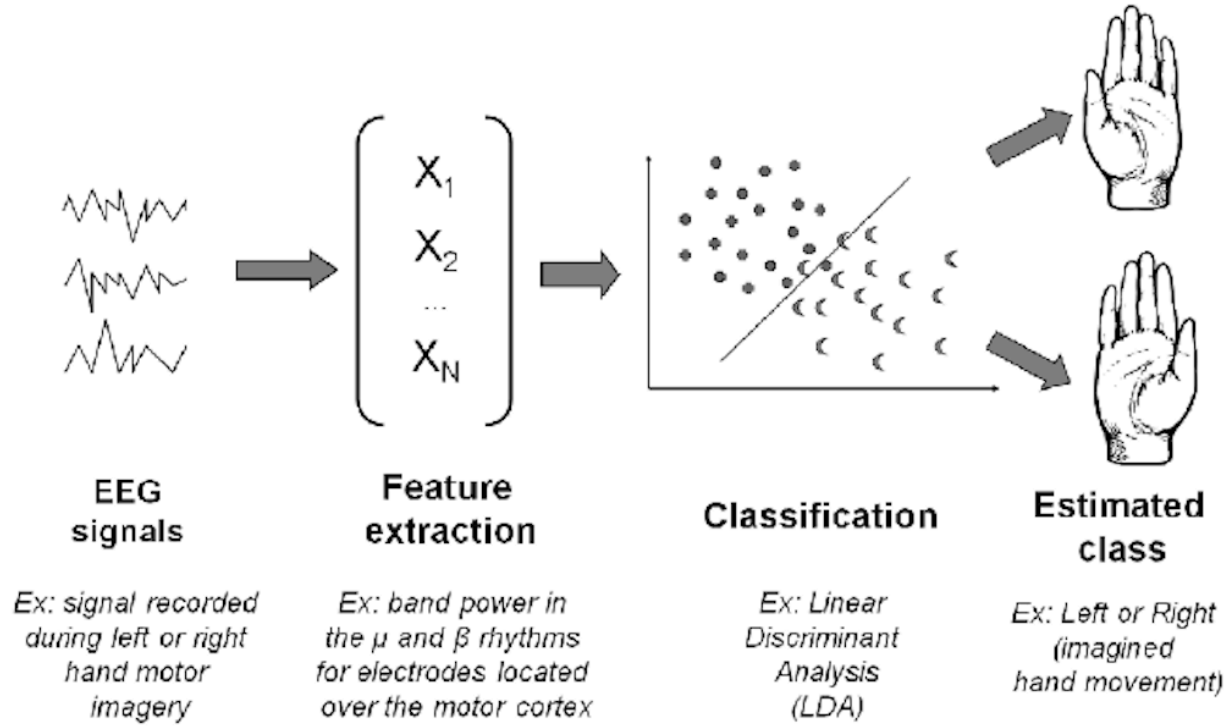
Berdasarkan contoh data di atas jika ingin melakukan prediksi harga rumah maka feature yang dapat digunakan adalah adalah luas tanah, luas bangunan, dan jumlah kamar tidur. Akan tetapi, feature ukuran torrent air tidak digunakan karena torrent air tidak akan mempengaruhi harga rumah.



Gambar 1. Fitur untuk Machine Learning (Sumber: Stanford)

2.2. Feature Extraction

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 sinyal EEG. Sinyal EEG (*electroencephalography*) 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 EEG. Setelah fitur didapat kemudian dilakukan klasifikasi yang kemudian menghasilkan prediksi pergerakan tangan kanan atau kiri.

2.3. 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 variabel-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 3000 meter 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 diharapkan dapat mengatasi performa yang buruk dari algoritma akibat skala variable yang berbeda.

Perbedaan skala untuk variabel input ini tidak mempengaruhi 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) dan 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 dan hasil output sebagai berikut:

```python
#example of a normalization
from numpy import asarray
from sklearn.preprocessing import MinMaxScale

#define data
data =  [[100,  0.001],
         [8, 0.05],
         [50, 0.005],
         [88, 0.07],
         [4, 0.1]]

print ("Data asli:")
print(data)

#define min max scaler
scaler = MinMaxScaler()

print ("Data  transform hasil normalisasi:")

#transform data
scaled = scaler.fit_transform(data)
print(scaled)
```

```
Data asli:
[[100, 0.001], [8, 0.05], [50, 0.005], [88, 0.07], [4, 0.1]]
Data  transform hasil normalisasi:
[[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 .

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

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

Dimana n adalah banyaknya data

Contoh :

Jika diketahui nilai rata-rata dari sekumpulan data adalah 10,0. Kemudian standar deviasinya adalah 5,0. Jika 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

Berikut merupakan implementasi dan hasil output dari standardisasi menggunakan **StandardScaler:**

```python
from numpy import asarray
from sklearn.preprocessing import StandardScale

#define data
data = ([[100, 0.001],
         [8, 0.05],
         [50, 0.005],
         [88, 0.07],
         [4, 0.1]])

print ("Data asli:")
print(data)

#define standard scaler
scaler = StandardScaler()

print ("Data  transform hasil standardisasi:")

#transform data
scaled = scaler.fit_transform(data)
print(scaled)
```

```
Data asli:
[[100, 0.001], [8, 0.05], [50, 0.005], [88, 0.07], [4, 0.1]]
Data  transform hasil normalisasi:
[[ 1.26398112 -1.16389967]
 [-1.06174414  0.12639634]
 [ 0.         -1.05856939]
 [ 0.96062565  0.65304778]
 [-1.16286263  1.44302493]]
```

2.4. Feature Extraction dari Data Kategorik

Dalam machine learning terdapat 2 jenis data yang sering digunakan yaitu 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 desimal. 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

A. ORDINAL ENCODING

Dalam ordinal encoding, setiap kategori yang unik diberi nilai integer. Misalnya, "merah" adalah 1, "hijau" adalah 2, dan "biru" adalah 3. Biasanya 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 seperti yang tertera pada tabel 1. Data tersebut akan diubah 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, 1, 2, dst sehingga didapatkan hasil pada tabel 2. Scikit-learn Python telah memiliki library untuk melakukan transformasi data kategorik ke data numerical yaitu Ordinal Encoder.

Tabel 1. Contoh data dengan tipe kategorik

Vocational College
State Polytechnics of Malang
Electronic State Polytechnics of Surabaya
State Polytechnics of Jakarta
State Polytechnics of Padang
State Polytechnics of Bandung

Tabel 2. Hasil pengurutan data vocational college

	Vocational College
0	Electronic State Polytechnics of Surabaya
1	State Polytechnics of Bandung
2	State Polytechnics of Jakarta
3	State Polytechnics of Malang
4	State Polytechnics of Padang

Contoh kode program dan hasil output :

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

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

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

```
[[3.]
 [0.]
 [2.]
 [4.]
 [1.]]
```

B. 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 penggunaaan 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 3 menunjukkan contoh one-hot encoding.

Tabel 3. Contoh penerapan One Hot Encoding pada data Vocational College

	0	**1**	**2**	**3**	**4**
Electronic State Polytechnics of Surabaya	1	0	0	0	0
State Polytechnics of Bandung	0	1	0	0	0
State Polytechnics of Jakarta	0	0	1	0	0
State Polytechnics of Malang	0	0	0	1	0
State Polytechnics of Padang	0	0	0	0	1

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 menunjukkan 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, 0, 0] dengan “1” untuk nilai biner pertama, kemudian State Polytechnics of Bandung direpresentasikan dengan [0, 1, 0, 0, 0], dan seterusnya.

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

```
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'},
    {'name' : 'State Polytechnics of Padang'},
    {'name' : 'State Polytechnics of Bandung'}
]

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

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

C. DUMMY VARIABLE ENCODING

One-hot encoding membuat satu variabel biner untuk setiap kategori. Terdapat redundansi dalam one-hot encoding. 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 "Electronic State Polytechnics of Surabaya", misal [0, 0]. Cara ini disebut variabel dummy encoding.

Tabel 4. Dummy Variable Encoding

Vocational College	Binary	
State Polytechnics of Malang	0	1
Electronic State Polytechnics of Surabaya	0	0
State Polytechnics of Jakarta	1	0

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 dan hasil output:

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

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

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

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

2.5. Feature Extraction 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 Menyusun sebuah kalimat.
- Stopword removal mengeliminasi kata – kata yang tidak penting berdasarkan daftar stopword. Contoh stopword adalah “yang”, “dan”, “di”, “dari” dan seterusnya.

Tabel 5. Contoh 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|

- Pada tahap stemming ini dilakukan untuk mencari kata dasar. Jadi, setiap kata yang memiliki imbuhan seperti imbuhan awalan dan akhiran, maka akan diambil kata dasarnya saja.
- 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 studi kasus TF-IDF:

1. Siapkan text dataset seperti contoh berikut.

```
corpus = [
            'the house had a tiny little mouse',
            'the cat saw the mouse',
            'the mouse ran away from the house',
            'the cat finally ate the mouse',
            'the end of the mouse story'
]
```

2. Pembobotan TF-IDF menggunakan TfidfVectorizer

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(stop_words='english')
response = vectorizer.fit_transform(corpus)
print(response)
```

Hasil output pembobotan TF-IDF:

```
(0, 7)        0.2808823162882302
(0, 6)        0.5894630806320427
(0, 11)       0.5894630806320427
(0, 5)        0.47557510189256375
(1, 9)        0.7297183669435993
(1, 2)        0.5887321837696324
(1, 7)        0.3477147117091919
(2, 1)        0.5894630806320427
(2, 8)        0.5894630806320427
(2, 7)        0.2808823162882302
(2, 5)        0.47557510189256375
(3, 0)        0.5894630806320427
(3, 4)        0.5894630806320427
(3, 2)        0.47557510189256375
(3, 7)        0.2808823162882302
(4, 10)       0.6700917930430479
(4, 3)        0.6700917930430479
(4, 7)        0.3193023297639811
```

Dimana kolom 1 merupakan Indeks dari dokumen pada corpus; kolom 2 merupakan Indeks dari token yang terdapat dalam kalimat tersebut, dan kolom 3 merupakan Bobot dari tf-idf hasil kalkukasi dari tf-idf vectorizer.

3. Gunakan vectorizer.get_feature_names_out() untuk menampilkan kumpulan token yang sudah dieliminasi stop-wordsnya, dan telah diurutkan sesuai abjad.

```
['ate' 'away' 'cat' 'end' 'finally' 'house' 'little' 'mouse' 'ran' 'saw'
 'story' 'tiny']
```

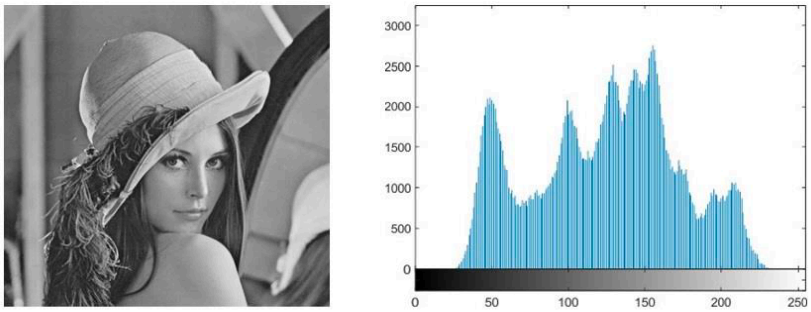
4. Transformasikan hasil response ke dalam bentuk array menggunakan response.todense()

	D1	D2	D3	D4	D5
ate	0.000000	0.000000	0.000000	0.589463	0.000000
away	0.000000	0.000000	0.589463	0.000000	0.000000
cat	0.000000	0.588732	0.000000	0.475575	0.000000
end	0.000000	0.000000	0.000000	0.000000	0.670092
finally	0.000000	0.000000	0.000000	0.589463	0.000000
house	0.475575	0.000000	0.475575	0.000000	0.000000
little	0.589463	0.000000	0.000000	0.000000	0.000000
mouse	0.280882	0.347715	0.280882	0.280882	0.319302
ran	0.000000	0.000000	0.589463	0.000000	0.000000
saw	0.000000	0.729718	0.000000	0.000000	0.000000
story	0.000000	0.000000	0.000000	0.000000	0.670092
tiny	0.589463	0.000000	0.000000	0.000000	0.000000

Ini merupakan nilai pembobotan yang sudah dinormalisasi dengan menggunakan L2 Normalization, nilai terkecilnya adalah 0,0 sedangkan nilai terbesarnya 1,0. Semakin tinggi bobot suatu kata terhadap suatu dokumen, mengindikasikan bahwa kata tersebut semakin layak untuk dijadikan keyword dari dokumen tersebut.

2.6. Feature Extraction pada Citra

Salah satu cara melalukan ekstraksi fitur citra adalah dengan menggunakan histogram. Histogram menunjukkan distribusi piksel berdasarkan intensitas gray level (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

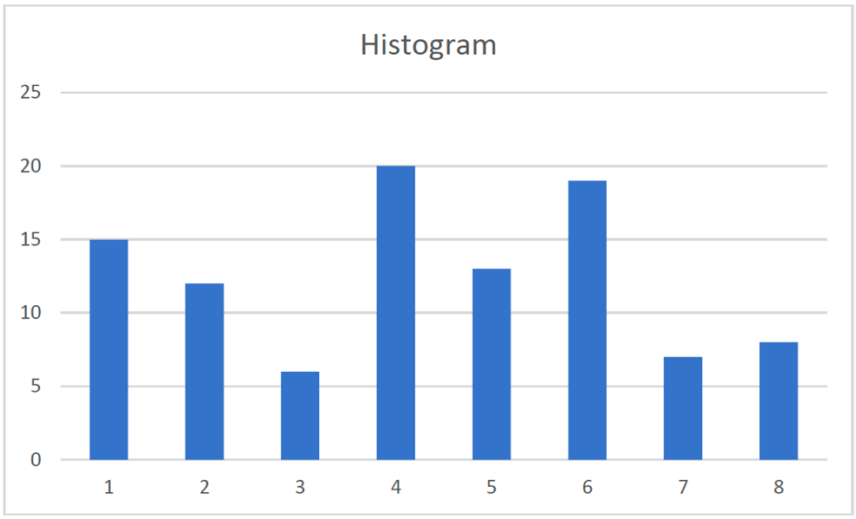
Contoh perhitungan histogram citra. Diketahui citra pada gambar dengan ukuran 10x10 dengan nilai pixel antara 0-7. Langkah pertama yang dilakukan adalah menghitung frekuensi kemunculan untuk setiap pixel seperti pada tabel 7 dimana x adalah nilai pixel dan y adalah jumlah kemunculan pixel (x). Selanjutnya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

Gambar 4. Citra dengan ukuran 10 x 10

Tabel 7. Tabel perhitungan frekuensi kemunculan setiap pixel

X	0	1	2	3	4	5	6	7
y	15	12	6	20	13	19	7	8



Gambar 5. Hasil Histogram

3. Tugas Praktikum

Buatlah program Python yang mengimplementasikan feature extraction sampai mendapatkan hasil akhir pembobotan TF-IDF menggunakan text dataset yang telah disiapkan!