Pemrosesan awal data merupakan langkah integral dalam Pembelajaran Mesin karena kualitas data dan informasi bermanfaat yang dapat diperoleh darinya secara langsung memengaruhi kemampuan model.

Konsep yang akan dibahas diantaranya :

- Handling Null Values
- Standardization
- Handling Categorical Variables
- One-Hot Encoding

```
# Importing necessary libraries
import numpy as np
import pandas as pd
```

## Handling Null Values

Dalam kumpulan data dunia nyata mana pun, selalu ada beberapa nilai nol. Tidak masalah apakah itu regresi, klasifikasi, atau jenis masalah lainnya, tidak ada model yang dapat menangani nilai NULL atau NaN ini sendiri sehingga perlu campur tangan.

```
# Creating a dummy dataframe
data = [['Steve',25,55],['Linus',28,60],['Elon',26]]
df = pd.DataFrame(data=data,columns=['Name','Age','Weight'])
```

```
# Displaying the dataframe
df
```

| | Name | Age | Weight |
|---|---|---|---|
| **0** | Steve | 25 | 55.0 |
| **1** | Linus | 28 | 60.0 |
| **2** | Elon | 26 | NaN |

```
# Checking null values
df.isnull()
```

| | Name | Age | Weight |
|---|---|---|---|
| **0** | False | False | False |
| **1** | False | False | False |
| **2** | False | False | True |

```
# Displaying the count of null values per column
df.isnull().sum()
```

```
Name      0
Age       0
Weight    1
dtype: int64
```

```
# Dropping the row having null value
# NOTE : The changes won't be reflected in the original dataframe until we set inplace=True
df.dropna()
```

| | Name | Age | Weight |
|---|---|---|---|
| **0** | Steve | 25 | 55.0 |
| **1** | Linus | 28 | 60.0 |

```
# Our original dataframe still has the 3rd row intact as we did not set inplace=True in the above operation
df
```

| | Name | Age | Weight |
|---|---|---|---|
| **0** | Steve | 25 | 55.0 |
| **1** | Linus | 28 | 60.0 |
| **2** | Elon | 26 | NaN |

## Imputation

Imputasi hanyalah proses mengganti nilai yang hilang dari kumpulan data. Kita dapat melakukannya dengan mendefinisikan fungsi kustom kita sendiri atau kita dapat melakukan imputasi dengan menggunakan kelas SimpleImputer yang disediakan oleh sklearn.

### Using the SimpleImputer class for Imputation

```
from sklearn.impute import SimpleImputer
```

```
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer = imputer.fit(df[['Weight']])
df['Weight'] = imputer.transform(df[['Weight']])
```

.**values** yang digunakan di sini mengembalikan representasi numpy dari bingkai data. Hanya nilai dalam bingkai data yang akan dikembalikan, label sumbu akan dihapus.

```
df
```

| | Name | Age | Weight |
|---|---|---|---|
| **0** | Steve | 25 | 55.0 |
| **1** | Linus | 28 | 60.0 |
| **2** | Elon | 26 | 57.5 |

### Using the fillna() method

```
df.fillna(df.mean())
```

```
<ipython-input-10-a2478f315f9e>:1: FutureWarning: Dropping of nuisance columns in
  df.fillna(df.mean())
```

| | Name | Age | Weight |
|---|---|---|---|
| **0** | Steve | 25 | 55.0 |
| **1** | Linus | 28 | 60.0 |
| **2** | Elon | 26 | 57.5 |

## Standardization

Ini adalah langkah preprocessing integral lainnya. Dalam Standardisasi, kita mengubah nilai sedemikian rupa sehingga rata-rata nilainya adalah 0 dan standar deviasinya adalah 1.

```
from sklearn.preprocessing import StandardScaler
```

Di sini kita memiliki 2 nilai numerik: Umur dan Berat. Nilai tersebut tidak berada pada skala yang sama dimana Umur dalam tahun dan Berat dalam Kg dan karena Berat cenderung lebih besar daripada Umur, oleh karena itu model akan memberi bobot lebih pada Berat, yang bukan merupakan skenario ideal karena Usia juga merupakan faktor integral di sini. Untuk menghindari masalah ini, kita melakukan Standardisasi.

```
# We have two numerical columns : Age and Weight, hence we will standardize them so that they are on the same scale
std = StandardScaler()
X = std.fit_transform(df[['Age','Weight']])
```

```
X
```

```
array([[-1.06904497, -1.22474487],
       [ 1.33630621,  1.22474487],
       [-0.26726124,  0.        ]])
```

Hal penting yang perlu diperhatikan di sini adalah kita perlu menstandarkan data pelatihan dan pengujian.

fit_transform sama dengan menggunakan fit lalu transform. fit function menghitung rata-rata dan standar deviasi dan fungsi transform benar-benar membakukan kumpulan data dan kita dapat melakukan proses ini dalam satu baris kode menggunakan fungsi fit_transform .

## Handling Categorical Variables

Menangani variabel kategori adalah aspek integral lainnya dari Machine Learning. Variabel kategori pada dasarnya adalah variabel yang diskrit dan tidak kontinu. Contoh - warna suatu barang adalah variabel diskrit sedangkan harganya adalah variabel kontinu.

Variabel kategori selanjutnya dibagi menjadi 2 jenis :

Variabel kategori ordinal — Variabel ini dapat diurutkan. Mis - Ukuran T-shirt. Bisa dibilang M<L<XL.

Variabel kategori nominal — Variabel ini tidak dapat dipesan. Mis - Warna T-shirt. Kami tidak dapat mengatakan bahwa Biru<Hijau karena tidak masuk akal untuk membandingkan warna karena tidak memiliki hubungan apa pun.

### Ordinal Categorical Variables

```
# Creating a dummy dataframe
df_cat = pd.DataFrame(data =
                      [['green','M',10.1,'class1'],
                       ['blue','L',20.1,'class2'],
                       ['white','M',30.1,'class1']])
df_cat.columns = ['color','size','price','classlabel']
```

```
df_cat
```

| | color | size | price | classlabel |
|---|---|---|---|---|
| **0** | green | M | 10.1 | class1 |
| **1** | blue | L | 20.1 | class2 |
| **2** | white | M | 30.1 | class1 |

### Using the map function

```
size_mapping = {'M':1,'L':2}
df_cat['size'] = df_cat['size'].map(size_mapping)
```

### Using Label Encoder

```
from sklearn.preprocessing import LabelEncoder
class_le = LabelEncoder()
df_cat['classlabel'] = class_le.fit_transform(df_cat['classlabel'].values)
```

Kesalahan terbesar yang dilakukan kebanyakan orang adalah mereka tidak dapat membedakan antara CV ordinal dan nominal. Jadi jika Anda menggunakan fungsi map() atau LabelEncoder yang sama dengan variabel nominal maka model akan berpikir bahwa ada semacam hubungan antara CV nominal.

### Nominal Categorical Variables

Cara penanganan CV nominal yang benar adalah dengan menggunakan One-Hot Encoding. Cara termudah untuk menggunakan One-Hot Encoding adalah dengan menggunakan fungsi get_dummies().

```
# Using One-Hot Encoding
df_cat = pd.get_dummies(df_cat[['color','size','price']])
```

```
df_cat
```

| | size | price | color_blue | color_green | color_white |
|---|---|---|---|---|---|
| **0** | 1 | 10.1 | 0 | 1 | 0 |
| **1** | 2 | 20.1 | 1 | 0 | 0 |
| **2** | 1 | 30.1 | 0 | 0 | 1 |