Modul Praktikum

Kecerdasan Buatan

4

Data Preprocessing

Pendahuluan

Data preprocessing adalah teknik yang digunakan untuk mengubah data mentah dalam format yang berguna dan efisien. Hal ini diperlukan karena data mentah seringkali tidak lengkap dan memiliki format yang tidak konsisten. Pengembangan sebuah AI memerlukan berbagai macam preprocessing sebelumnya untuk memberikan hasil yang andal, tepat, dan akurat

Setup Environment

Dalam melakukan *preprocessing*, kita akan menggunakan beberapa library yang terdapat di Python. Berikut beberapa library yang akan kita pakai:

1. **Pandas**

   Pandas adalah *open source python package* yang paling banyak digunakan dalam data science, data analysis, dan machine learning. Dimana Pandas dibangun di atas *python package* lain bernama Numpy, yang menyediakan dukungan untuk array multidimensi. Dokumentasi pandas dapat dilihat di link ini. Untuk menginstall pandas buka CMD(Win + R, ketik “cmd”, Enter) >> `pip install pandas`

2. **Numpy**

   NumPy merupakan salah satu library Python yang berfungsi untuk proses komputasi numerik. NumPy memiliki kemampuan untuk membuat objek berdimensi array. Array merupakan sekumpulan variabel yang memiliki tipe data yang sama. Kelebihan dari NumPy adalah dapat memudahkan operasi komputasi pada data, cocok untuk melakukan akses secara acak, dan elemen array merupakan sebuah nilai yang independen sehingga penyimpanannya dianggap sangat efisien. Dokumentasi pandas dapat dilihat di link ini. Untuk menginstall numpy buka CMD(Win + R, ketik “cmd”, Enter) >> `pip install numpy`

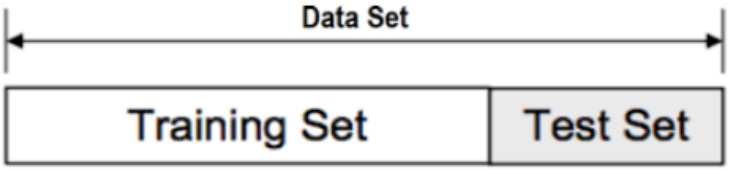
3. **Scikit - Learn**

   Scikit-learn atau sklearn merupakan sebuah module dari bahasa pemrograman Python yang dibangun berdasarkan NumPy, SciPy, dan Matplotlib. Fungsi dari modul ini adalah untuk membantu melakukan kebutuhan machine learning seperti seperti regresi (regression), klasifikasi (classification), pengelompokkan/penggugusan (clustering), data

preprocessing, dimensionality reduction, dan model selection (perbandingan, validasi, dan pemilihan parameter maupun model).Dokumentasi scikit learn dapat dilihat di link ini. Untuk menginstall sklearn buka CMD(Win + R, ketik "cmd", Enter) >> `pip install sklearn`

Data Split

Train/test split adalah salah satu metode yang dapat digunakan untuk mengevaluasi performa model machine learning. Metode evaluasi model ini membagi dataset menjadi dua bagian yakni bagian yang digunakan untuk *training data* dan untuk *testing data* dengan proporsi tertentu.



Train data digunakan untuk fit model machine learning, sedangkan *test data* digunakan untuk mengevaluasi hasil fit model tersebut.Untuk menggunakannya, kita perlu mengimport Scikit-Learn terlebih dahulu, kemudian setelah itu kita dapat menggunakan fungsi train_test_split().

```
from sklearn.model_selection import train_test_split
```

Contoh :

```
# Import module yang dibutuhkan

from sklearn.model_selection import train_test_split

import pandas as pd
```

```
# Baca data dan ditampung dalam variabel

df = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv')
```

```
# Subset feature dan target

X = df.iloc[:,:-1] # Target

y = df.iloc[:,-1] # Feature
```

```
# Buat kelas train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
print("Dimensi X_train : ", X_train.shape)

print("Dimensi X_train : ", X_test.shape)

print("Dimensi y_train : ", y_train.shape)

print("Dimensi y_test : ", y_test.shape)
```

Output :

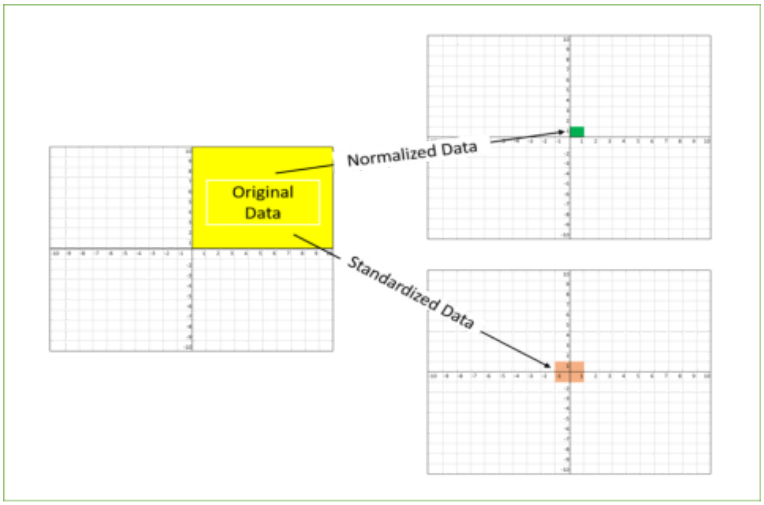
```
Dimensi X_train :  (120, 4)
Dimensi X_train :  (30, 4)
Dimensi y_train :  (120,)
Dimensi y_test :  (30,)
```

- **X_train** : Untuk menampung data source yang akan dilatih.
- **X_test** : Untuk menampung data target yang akan dilatih.
- **y_train** : Untuk menampung data source yang akan digunakan untuk testing.
- **y_test** : Untuk menampung data target yang akan digunakan untuk testing.
- **test_size** : Mendefinisikan ukuran testing set (0.2 = 20% dari seluruh dataset)

Data Transforming

Transformasi data adalah merubah skala data kedalam bentuk lain sehingga data memiliki distribusi yang diharapkan.Scaling adalah suatu cara untuk membuat numerical data pada dataset memiliki rentang nilai (scale) yang sama. Tidak ada lagi satu variabel data yang mendominasi variabel data lainnya.Teknik penskalaan yang biasa dipakai yaitu normalisasi dan standardisasi

- **Normalisasi** adalah proses penskalaan data menjadi rentang [0, 1].
- **Standardisasi** adalah proses penskalaan data sehingga memiliki nilai rata-rata 0 dan standar deviasi 1.



Untuk melakukan standardisasi, Kita dapat menggunakan StandardScaler pada scikit learn

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

Contoh:

```
# Import module yang dibutuhkan

from sklearn.preprocessing import StandardScaler

import numpy as np
```

```
# Buat ‘dummy’ data

data = np.asarray([[100, 0.001],

                   [8, 0.05],

                   [50, 0.005],

                   [88, 0.07],])
```

```
# Buat obyek StandardScaler lalu lakukan perubahan pada data menggunakan metode ‘fit_transform’ pada data

ss = StandardScaler()

Scaled_data = ss.fit_transform(data)
```

```
print("Nilai Data Sebelum scaling : ")

print(data)

print("Nilai standar deviasi: ", np.std(data))



print("\n Nilai Data setelah scaling : ")

print(Scaled_data)

print("Nilai standar deviasi: ", np.std(Scaled_data))
```

Output :

```
Nilai Data Sebelum scaling :
[[1.0e+02 1.0e-03]
 [8.0e+00 5.0e-02]
 [5.0e+01 5.0e-03]
 [8.8e+01 7.0e-02]]
Nilai standar deviasi:  39.89949316955668

Nilai Data setelah scaling :
[[ 1.06996056 -1.03748098]
 [-1.48682831  0.62929174]
 [-0.31959861 -0.9014179 ]
 [ 0.73646636  1.30960714]]
Nilai standar deviasi:  1.0
```

Data Cleaning

Data yang kita gunakan bisa jadi sangat berantakan. Ada informasi yang tak lengkap, ada pula format yang berbeda-beda.Semua ini tentu membuat data itu sulit diolah. Pada akhirnya, ia jadi tak bisa dimanfaatkan. Untuk itulah kita harus melakukan data cleaning terlebih dahulu. Data cleaning adalah memperbarui, membetulkan, dan membuat data menjadi lebih rapi. Beberapa model machine learning tidak dapat bekerja dengan data yang masih berantakan. Kesalahan dalam data dapat berupa:

- Terdapat nilai data yang tidak valid (Null)
- Data dalam format yang salah
- Terdapat duplikat

1. Menangani nilai Null

Null adalah kata kunci (keyword) khusus yang berarti tidak memiliki nilai.Banyak algoritma machine learning yang tidak dapat bekerja jika terdapat nilai **null** dalam data. Berikut beberapa cara untuk mengatasi nilai null:

- **Menghapus record/feature** : Kita dapat menghapus record / feature yang memiliki nilai null namun cara ini dapat menghilangkan informasi - informasi yang berguna
- **Subtitusi nilai :** Kita dapat mengganti nilai null dengan suatu metrik pengukuran seperti mean atau median dari kolom data itu.

Misal, kita mempunyai sebuah dataframe yang memiliki beberapa nilai null:

	Nama	Jurusan	IPK
0	Lusi	Informatika	2.5
1	Aisyah	Kimia	3.2
2	Rizky	Sipil	NaN
3	Dendi	Kimia	2.5
4	Bagus	Informatika	3.1
5	Ajeng	Elektro	3.8
6	Putri	Industri	3.1
7	Yusuf	Sipil	NaN
8	Kulus	Kimia	3.8
9	Lusi	Informatika	2.5

Untuk menangani nilai null pada data, kita dapat menggunakan kelas SimpleImputer pada scikit - learn

```
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy='mean' )
```

Ketika membuat objek dari kelas SimpleImputer, terdapat parameter yang harus diatur:

- **strategy :** berfungsi untuk mengganti nilai data yang kosong (null) dengan rata - rata kolom (‘mean’), bisa juga menggunakan ‘median’, ‘most_frequent’ (modus), atau ‘constant’

Setelah obyek imputer telah dibuat, kita dapat menggunakan fungsi fit_transform() untuk mengubah nilai null pada kolom yang kita tentukan berdasarkan strategy yang digunakan pada kelas SimpleImputer

```
df["IPK"] = imputer.fit_transform(df[["IPK"]])
```

Dataframe yang sudah diperbaharui terlihat seperti ini:

	Nama	Jurusan	IPK
0	Lusi	Informatika	2.5000
1	Aisyah	Kimia	3.2000
2	Rizky	Sipil	3.0625
3	Dendi	Kimia	2.5000
4	Bagus	Informatika	3.1000
5	Ajeng	Elektro	3.8000
6	Putri	Industri	3.1000
7	Yusuf	Sipil	3.0625
8	Kulus	Kimia	3.8000
9	Lusi	Informatika	2.5000

Contoh:

```
# Import module yang dibutuhkan

import numpy as np

import pandas as pd

from sklearn.impute import SimpleImputer
```

```
# Buat item untuk dataframe

Nama                                                                    =
["Lusi","Aisyah","Rizky","Dendi","Bagus","Ajeng","Putri","Yusuf","Kulus","L
usi"]
```

```
Jurusan                                                                    =
["Informatika","Kimia","Sipil","Kimia","Informatika","Elektro","Industri","
Sipil","Kimia","Informatika"]

IPK = [2.5,3.2,np.nan,2.5,3.1,3.8,3.1,np.nan,3.8,2.5]

# Buat dataframe

df = pd.DataFrame(

   {

       "Nama":Nama,

       "Jurusan" : Jurusan,

       "IPK" : IPK

   }

)
```

```
print("Jumlah record yang memiliki nilai null: ")

print(df.isna().sum())
```

```
#Deklarasi kelas SimpleImputer dan lakukan transformasi pada kolom yang
ingin diubah

imputer = SimpleImputer(strategy='mean')

df["IPK"] = imputer.fit_transform(df[["IPK"]])
```

```
print("\nJumlah null setelah menggunakan SimpleImputer: ")
```

```
print(df.isna().sum())
```

Output:

```
Jumlah record yang memiliki nilai null:
Nama       0
Jurusan    0
IPK        2
dtype: int64

Jumlah null setelah menggunakan fungsi fillna():
Nama       0
Jurusan    0
IPK        0
dtype: int64
```

Kita juga dapat mengganti nilai **null** dalam data menggunakan fungsi bawaan dari pandas yaitu `fillna()`.

Contoh :

```
IPK = [2.5,3.2,np.nan,2.5,3.1,3.8,3.1,np.nan,3.8,2.5]

df["IPK"] = IPK

df.isna().any()
```

Output:

```
Nama       False
Jurusan    False
IPK         True
dtype: bool
```

```
#Gunakan fungsi fillna pada kolom yang memiliki nilai null

df["IPK"].fillna(df["IPK"].mean(),inplace=True)

df.isna().any()
```

Output:

```
Nama       False
Jurusan    False
IPK        False
dtype: bool
```

Kita juga bisa menghapus sebuah kolom untuk mengatasi nilai **null** dalam data dengan menggunakan fungsi bawaan pandas dropna() . Cara ini biasa dilakukan ketika suatu kolom mempunyai nilai null yang sangat banyak

```
IPK = [2.5,3.2,np.nan,2.5,3.1,3.8,3.1,np.nan,3.8,2.5]

df["IPK"] = IPK

df.head()
```

Output :

	Nama	Jurusan	IPK
0	Lusi	Informatika	2.5
1	Aisyah	Kimia	3.2
2	Rizky	Sipil	NaN
3	Dendi	Kimia	2.5
4	Bagus	Informatika	3.1

```
#Gunakan metode dropna dari pandas untuk menghapus kolom yang memiliki nilai null

df.dropna(inplace = True)

df
```

Output :

	Nama	Jurusan
0	Lusi	Informatika
1	Aisyah	Kimia
2	Rizky	Sipil
3	Dendi	Kimia
4	Bagus	Informatika

2. Menangani nilai duplikat

Salah satu langkah dalam membersihkan suatu data adalah dengan menghapus nilai duplikat. Nilai duplikat dapat membuat sebaran data condong kedalam suatu nilai yang dapat mengacaukan hasil analisis kita. Menghapus nilai duplikat juga dapat mengurangi beban komputasi kita. Untuk menghapus nilai duplikat kita dapat menggunakan fungsi dari pandas yaitu drop_duplicates

Contoh :

```
# Import module

import pandas as pd
```

```
# Buat item untuk dataframe

Nama                                                                    =
["Lusi","Aisyah","Rizky","Dendi","Bagus","Ajeng","Putri","Yusuf","Kulus","L
usi"]

IPK = [2.5,3.2,2.5,2.5,3.1,3.8,3.1,2.4,3.8,2.5]



# Buat dataframe

df = pd.DataFrame(

   {

       "Nama":Nama,

       "IPK" : IPK

   }

)
```

```
print("Nilai Duplikat: ")

df[df.duplicated()]
```

Output :

Nilai Duplikat:

	Nama	IPK
9	Lusi	2.5

```
# Hapus duplikat menggunakan metode drop_duplicates dari pandas

df.drop_duplicates(inplace=True)
```

```
print("\nSetelah menggunakan fungsi drop_duplicates()")

print("Jumlah Nilai Duplikat:",df.duplicated().sum())
```

Output :

```
Setelah menggunakan fungsi drop_duplicates()
Jumlah Nilai Duplikat: 0
```

Encoding

Encoding adalah salah tahap *pre-processing* ketika kita menghadapi data yang bersifat kategorikal. **Encoding** adalah teknik yang mengubah variabel kategorikal menjadi variabel numerik, model dalam machine learning membutuhkan input yang bertipe numerik sehingga sangat penting bagi kita untuk melakukan **encoding** jika terdapat variabel yang bertipe kategorikal didalam data kita. Teknik **encoding** yang paling terkenal adalah **Ordinal Encoding** dan **One-Hot Encoding**

1. **Ordinal Encoding**

   Dalam **ordinal encoding**, setiap nilai kategori yang unik akan diubah menjadi nilai integer. Tipe **encoding** ini biasa digunakan jika terdapat unsur hirarki di dalam variabel yang ingin diubah. Contohnya adalah ketika nilai data berupa Low,Medium,High (“Low” = 1 , “Medium” = 2, “High” = 3)

   Untuk melakukan **ordinal encoding** kita dapat menggunakan fungsi dari scikit Learn OrdinalEncoder

```
from sklearn.preprocessing import OrdinalEncoder
```

Contoh :

```
# Import module

import pandas as pd

from sklearn.preprocessing import OrdinalEncoder
```

```
# Buat item untuk dataframe

Nama                                                                    =
["Lusi","Aisyah","Rizky","Dendi","Bagus","Ajeng","Putri","Yusuf","Kulus","L
usi"]

Jurusan                                                                 =
["Informatika","Kimia","Sipil","Kimia","Informatika","Elektro","Industri","
Sipil","Kimia","Informatika"]



# Buat dataframe

df = pd.DataFrame(

   {

       "Nama":Nama,

       "Jurusan" : Jurusan,

   }
```

```
#Deklarasi kelas OrdinalEncoder dan lakukan transformasi pada kolom yang
ingin diubah

encoder = OrdinalEncoder()

df["JurusanEnc"] = encoder.fit_transform(df[["Jurusan"]])

df.head()
```

Output :

	Nama	Jurusan	JurusanEnc
0	Lusi	Informatika	2.0
1	Aisyah	Kimia	3.0
2	Rizky	Sipil	4.0
3	Dendi	Kimia	3.0
4	Bagus	Informatika	2.0

2. One - Hot Encoding

One-Hot encoding adalah salah satu metode **encoding**. Metode ini merepresentasikan data bertipe kategori sebagai vektor biner yang bernilai integer, 0 dan 1, dimana semua elemen akan bernilai 0 kecuali satu elemen yang bernilai 1, yaitu elemen yang memiliki nilai kategori tersebut.

Kita dapat menggunakan OneHotEncoder dari Scikit-Learn untuk menerapkan one-hot encoding di Python

```
from sklearn.preprocessing import OneHotEncoder
```

Contoh :

```
# Import module

import pandas as pd

from sklearn.preprocessing import OneHotEncoder
```

```
# Buat item untuk dataframe

Nama                                                                    =
["Lusi","Aisyah","Rizky","Dendi","Bagus","Ajeng","Putri","Yusuf","Kulus","L
usi"]

Jurusan                                                                 =
["Informatika","Kimia","Sipil","Kimia","Informatika","Elektro","Industri","
Sipil","Kimia","Informatika"]



# Buat dataframe

df = pd.DataFrame({

       "Nama":Nama,

       "Jurusan" : Jurusan,}

)
```

```
#Deklarasi kelas OneHotEncoder dan lakukan transformasi pada kolom yang ingin diubah

encoder = OneHotEncoder(sparse=False)

jurusanEnc = encoder.fit_transform(df[["Jurusan"]])
```

```
jurusanEnc = pd.DataFrame(jurusanEnc)

df = df.join(jurusanEnc)

df
```

Output:

	Nama	Jurusan	0	1	2	3	4
0	Lusi	Informatika	0.0	0.0	1.0	0.0	0.0
1	Aisyah	Kimia	0.0	0.0	0.0	1.0	0.0
2	Rizky	Sipil	0.0	0.0	0.0	0.0	1.0
3	Dendi	Kimia	0.0	0.0	0.0	1.0	0.0
4	Bagus	Informatika	0.0	0.0	1.0	0.0	0.0
5	Ajeng	Elektro	1.0	0.0	0.0	0.0	0.0
6	Putri	Industri	0.0	1.0	0.0	0.0	0.0
7	Yusuf	Sipil	0.0	0.0	0.0	0.0	1.0
8	Kulus	Kimia	0.0	0.0	0.0	1.0	0.0
9	Lusi	Informatika	0.0	0.0	1.0	0.0	0.0

Kita juga dapat menggunakan get_dummies dari Pandas untuk menerapkan **one-hot encoding**

Contoh :

```
# Import module

import pandas as pd
```

```
Nama                                                                    =
["Lusi","Aisyah","Rizky","Dendi","Bagus","Ajeng","Putri","Yusuf","Kulus","L
usi"]

Jurusan                                                                 =
["Informatika","Kimia","Sipil","Kimia","Informatika","Elektro","Industri","
Sipil","Kimia","Informatika"]



# Buat dataframe

df = pd.DataFrame({

       "Nama":Nama,

       "Jurusan" : Jurusan,}

)
```

```
enc = pd.get_dummies(df[["Jurusan"]])

jurusanEnc = pd.DataFrame(enc)
```

```
df = df.join(jurusanEnc)

df.head()
```

Output:

	Nama	Jurusan	Jurusan_Elektro	Jurusan_Industri	Jurusan_Informatika	Jurusan_Kimia	Jurusan_Sipil
0	Lusi	Informatika	0	0	1	0	0
1	Aisyah	Kimia	0	0	0	1	0
2	Rizky	Sipil	0	0	0	0	1
3	Dendi	Kimia	0	0	0	1	0
4	Bagus	Informatika	0	0	1	0	0

Source

1. **Data Split**
    - Evaluasi Model Machine Learning: Train/Test Split
2. **Data Transforming**
    - Why Data Scaling is important in Machine Learning & How to effectively do it
    - Transformasi Data Mining
    - Melakukan Feature Scaling pada Dataset
    - All about Feature Scaling
3. **Data Cleaning**
    - Mengenal Data Cleansing
    - Imputing Missing Values using the SimpleImputer Class in sklearn
4. **Encoding**
    - Data Preprocessing pada Machine Learning : Handling Categorical Data