

JOBSHEET 7

Data Pre-processing

A. TUJUAN

1. Mahasiswa dapat melakukan preparation data dengan menggunakan One Hot Encoding
2. Mahasiswa dapat melakukan preparation data dengan menggunakan normalization dan standardization
3. Mahasiswa memahami cara membagi data menjadi data training dan data testing

B. PETUNJUK

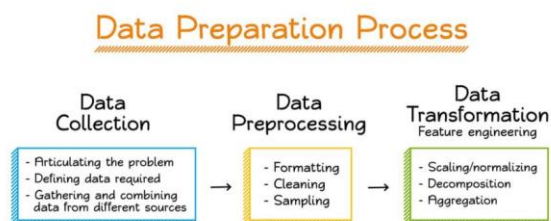
Awali setiap aktivitas dengan doa, semoga berkah dan mendapat kemudahan. Pahami tujuan, dasar teori, dan latihan-latihan praktikum dengan baik dan benar. Kerjakan tugas-tugas praktikum dengan baik, sabar, dan jujur. Tanyakan kepada asisten/dosen apabila ada hal-hal yang kurang jelas.

C. DASAR TEORI

1. Data Preparation

Data preparation adalah serangkaian langkah untuk membersihkan, mengorganisir, dan mengubah data mentah agar siap digunakan dalam analisis atau pemodelan. Tugas-tugas dalam data preparation meliputi pembersihan data, transformasi data, integrasi data, reduksi dimensi, penghapusan noise, dan pengaturan struktur data.

Tujuannya adalah memastikan kualitas data yang baik sehingga hasil analisis dapat akurat dan dapat diandalkan.



2. Dataset

Dataset adalah kumpulan data terstruktur yang digunakan untuk analisis, pemodelan, atau penelitian. Ini bisa berupa angka, teks, gambar, suara, atau kombinasi dari semuanya. Dataset memiliki atribut atau kolom yang mewakili variabel yang terkait. Mereka berasal dari berbagai sumber dan bisa dalam format file seperti CSV, Excel, atau database. Dataset penting untuk menemukan pola, mendapatkan wawasan, atau membangun model prediktif. Mereka memerlukan pemrosesan seperti pembersihan, transformasi, dan integrasi untuk memastikan kualitas dan relevansi data.

D. LATIHAN

1. Data Preparation dengan One Hot Encoding

One Hot Encoding mengubah data kategorik dengan membuat kolom baru untuk setiap kategori dibawah ini merupakan contoh implementasi dalam python

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
```

```
# membuat data array gender
gender = ['perempuan', 'laki-laki', 'laki-laki',
          'perempuan', 'perempuan']

# membuat data frame dari data gender
df = pd.DataFrame({'gender': gender})
```

```
# membuat instance dari OneHotEncoder
encoder = OneHotEncoder()

# melakukan encoding pada data frame
encoded_df =
pd.DataFrame(encoder.fit_transform(df[['gender']]).toarray(),
             columns=encoder.get_feature_names(['gender']))
```

```
# menggabungkan data frame yang sudah di-encode dengan
data frame awal
df_encoded = pd.concat([df, encoded_df], axis=1)

# menampilkan hasil
df_encoded
```

```
# menggabungkan data frame yang sudah di-encode dengan data frame awal
df_encoded = pd.concat([df, encoded_df], axis=1)

# menampilkan hasil
df_encoded
```

[5] ✓ 0.1s

	gender	gender_laki-laki	gender_perempuan
0	perempuan	0.0	1.0
1	laki-laki	1.0	0.0
2	laki-laki	1.0	0.0
3	perempuan	0.0	1.0
4	perempuan	0.0	1.0

Pada contoh kode di atas, pertama-tama kita membuat array gender yang berisi data gender. Kemudian, kita membuat data frame df dari array tersebut. Selanjutnya, kita membuat instance dari OneHotEncoder.

Selanjutnya, kita menggunakan fit_transform dari OneHotEncoder untuk melakukan encoding pada kolom gender pada data frame df. Hasil encoding kemudian disimpan pada data frame encoded_df.

Kita kemudian menggabungkan data frame df dengan data frame encoded_df menggunakan pd.concat, dan menyimpan hasilnya pada df_encoded. Terakhir, kita mencetak hasilnya dengan menggunakan print(df_encoded).

Jalankan kode program diatas menggunakan Jupyter Notebook atau Google Colab. Kemudian masukan hasil luaran dari kode diatas pada laporan praktikum !

2. Data Preparation dengan Outlier Removal

Outlier removal adalah teknik penghapusan data yang berada jauh dari sebagian besar data dalam sebuah dataset. Hal ini biasanya dilakukan untuk menghilangkan nilai yang sangat tidak biasa atau tidak mewakili

dari data yang dimiliki, yang dapat mempengaruhi analisis dan prediksi yang dihasilkan.

Berikut ini adalah contoh outlier removal pada sebuah dataframe dengan menggunakan library pandas dan NumPy

```
import pandas as pd
import numpy as np
```

```
# membuat data frame contoh
df = pd.DataFrame({
    'A': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'B': [15, 20, 25, 30, 35, 40, 45, 50, 55, 500]
})
```

```
# mencari nilai z-score untuk setiap data point pada
kolom B
z_scores = np.abs((df['B'] - df['B'].mean()) /
df['B'].std())

# menghilangkan data point yang memiliki z-score lebih
besar dari 3
df_clean = df.loc[round(z_scores) < 3]

# menampilkan data frame yang telah di-clean
print(df_clean)
```

```
▶ # mencari nilai z-score untuk setiap data point pada kolom B
z_scores = np.abs((df['B'] - df['B'].mean()) / df['B'].std())

# menghilangkan data point yang memiliki z-score lebih besar dari 3
df_clean = df.loc[round(z_scores) < 3]

# menampilkan data frame yang telah di-clean
print(df_clean)
```

[8] ✓ 0.1s

	A	B
0	1	15
1	2	20
2	3	25
3	4	30
4	5	35
5	6	40
6	7	45
7	8	50
8	9	55

Pada contoh kode di atas, kita membuat sebuah data frame contoh dengan dua kolom yaitu A dan B. Kolom B memiliki sebuah outlier dengan nilai 500 yang jauh lebih besar dari sebagian besar data.

Untuk menghilangkan outlier tersebut, kita pertama-tama menghitung nilai z-score untuk setiap data point pada kolom B menggunakan formula $(x - \text{mean}) / \text{std}$. Kemudian, kita menggunakan nilai z-score tersebut untuk menghilangkan data point yang memiliki z-score lebih besar dari 3 (yaitu data point yang jauh dari sebagian besar data).

Jalankan kode program diatas menggunakan Jupyter Notebook atau Google Colab. Kemudian masukan hasil luaran dari kode diatas pada laporan praktikum !

3. Data Preparation dengan Normalization

Normalization adalah salah satu teknik yang dipakai dalam data preparation. Tujuan dari normalisasi adalah mengubah nilai-nilai dari

sebuah fitur ke dalam skala yang sama. Normalization memungkinkan kenaikan performa dan stabilitas dari sebuah model machine learning.

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
```

```
# membuat data frame contoh
df = pd.DataFrame({
    'Nama': ['Andi', 'Budi', 'Cindy', 'Diana', 'Eka',
            'lala'],
    'Gaji': [3000000, 5000000, 7000000, 9000000,
            11000000, 9000000],
    'Umur': [25, 30, 35, 40, 45, 30]
})
```

```
# normalisasi data gaji dan umur menggunakan
MinMaxScaler
scaler = MinMaxScaler()
df[['Gaji', 'Umur']] = scaler.fit_transform(df[['Gaji',
            'Umur']])

# menampilkan data frame hasil normalisasi
df
```

```
# normalisasi data gaji dan umur menggunakan MinMaxScaler
scaler = MinMaxScaler()
df[['Gaji', 'Umur']] = scaler.fit_transform(df[['Gaji', 'Umur']])

# menampilkan data frame hasil normalisasi
df
```

[11] ✓ 0.1s

	Nama	Gaji	Umur
0	Andi	0.00	0.00
1	Budi	0.25	0.25
2	Cindy	0.50	0.50
3	Diana	0.75	0.75
4	Eka	1.00	1.00
5	lala	0.75	0.25

Pada contoh kode di atas, kita membuat sebuah data frame contoh dengan tiga kolom yaitu Nama, Gaji, dan Umur. Kolom Gaji dan Umur memiliki skala yang berbeda, sehingga perlu dilakukan normalisasi untuk membandingkan data tersebut secara akurat.

Kita menggunakan library scikit-learn untuk melakukan normalisasi menggunakan MinMaxScaler, di mana skala data akan diubah menjadi rentang 0-1. Kemudian, kita mengaplikasikan normalisasi pada kolom Gaji dan Umur dengan menggunakan method `fit_transform()`.

Jalankan kode program diatas menggunakan Jupyter Notebook atau Google Colab. Kemudian masukan hasil luaran dari kode diatas pada laporan praktikum !

4. Data Preparation dengan Standarization

Standardization adalah proses konversi nilai-nilai dari suatu fitur sehingga nilai-nilai tersebut memiliki skala yang sama. Z score adalah metode paling populer untuk standardisasi di mana setiap nilai pada sebuah atribut numerik akan dikurangi dengan rata-rata dan dibagi dengan standar deviasi dari seluruh nilai pada sebuah kolom atribut.

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
```

```
# membuat data frame contoh
df = pd.DataFrame({
    'Nama': ['Andi', 'Budi', 'Cindy', 'Diana', 'Eka',
            'lala'],
    'Gaji': [3000000, 5000000, 7000000, 9000000,
            11000000, 9000000],
    'Umur': [25, 30, 35, 40, 45, 30]
})
```

```
# normalisasi data gaji dan umur menggunakan
MinMaxScaler
scaler = MinMaxScaler()
df[['Gaji', 'Umur']] = scaler.fit_transform(df[['Gaji',
            'Umur']])

# menampilkan data frame hasil normalisasi
df
```

```
# normalisasi data gaji dan umur menggunakan MinMaxScaler
scaler = MinMaxScaler()
df[['Gaji', 'Umur']] = scaler.fit_transform(df[['Gaji', 'Umur']])

# menampilkan data frame hasil normalisasi
df
```

[14] ✓ 0.1s

	Nama	Gaji	Umur
0	Andi	0.00	0.00
1	Budi	0.25	0.25
2	Cindy	0.50	0.50
3	Diana	0.75	0.75
4	Eka	1.00	1.00
5	Iala	0.75	0.25

Pada contoh kode di atas, kita juga membuat sebuah data frame contoh dengan tiga kolom yaitu Nama, Gaji, dan Umur. Kita menggunakan library scikit-learn untuk melakukan normalisasi menggunakan StandardScaler, di mana data akan diubah menjadi memiliki mean=0 dan variance=1. Kemudian, kita mengaplikasikan normalisasi pada kolom Gaji dan Umur dengan menggunakan method `fit_transform()`.

Jalankan kode program diatas menggunakan Jupyter Notebook atau Google Colab. Kemudian masukan hasil luaran dari kode diatas pada laporan praktikum !

5. Pembuatan DataSet

Dataset yang telah dibersihkan dan diproses kemudian siap kita latih dengan machine learning. Satu-satunya cara untuk mengetahui apakah model machine learning kita bagus atau tidak adalah dengan mengujinya pada kasus atau data baru yang belum dikenali oleh model. Oleh karena itu kita perlu membagi dataset mejadi 2 bagian yaitu data training dan datatesting. Berikut adalah gambaran cara membagi dataset mejadi training set dan test set.

```
from sklearn.model_selection import train_test_split
import pandas as pd
```

```
# membuat data frame contoh
df = pd.DataFrame({
    'X': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Y': [15, 20, 25, 30, 35, 40, 45, 50, 55, 60]
})
```

```
# membagi data menjadi data latih dan data uji dengan
rasio 80:20
X_train, X_test, y_train, y_test =
train_test_split(df[['X']], df[['Y']], test_size=0.2,
random_state=42)
```

```
# menampilkan data latih
print('Data Latih')
print('X_train:', X_train)
print('y_train:', y_train)

# menampilkan data uji
print('\nData Uji')
print('X_test:', X_test)
print('y_test:', y_test)
```

```
▶ # menampilkan data latih
print('Data Latih')
print('X_train:', X_train)
print('y_train:', y_train)

# menampilkan data uji
print('\nData Uji')
print('X_test:', X_test)
print('y_test:', y_test)
```

[18] ✓ 0.0s

```
... Data Latih
X_train:      X
5    6
0    1
7    8
2    3
9   10
4    5
3    4
6    7
y_train:      Y
5   40
0   15
7   50
2   25
9   60
4   35
3   30
6   45

Data Uji
X_test:      X
8    9
1    2
y_test:      Y
8   55
1   20
```

Pada kode di atas, kita menggunakan fungsi `train_test_split` dari library `scikit-learn` untuk membagi data menjadi data latih dan data uji. Pertama-tama, kita membuat sebuah data frame contoh dengan dua kolom yaitu X dan Y. Selanjutnya, kita memanggil fungsi `train_test_split` dengan memasukkan kolom X sebagai fitur dan kolom Y sebagai target. Kita juga memasukkan argumen `test_size=0.2` yang artinya kita ingin membagi data menjadi data latih sebanyak 80% dan data uji sebanyak 20% dari keseluruhan data. Argumen `random_state` digunakan untuk menghasilkan hasil acak yang konsisten pada setiap eksekusi program

Implementasi Pembagian data pada data [heart.csv](#)

```
from sklearn.model_selection import train_test_split
import pandas as pd
```

```
# Import data heart.csv
df = pd.read_csv('heart.csv');
df
```

```
from sklearn.model_selection import train_test_split
import pandas as pd

# Import data heart.csv
df = pd.read_csv('heart.csv');
df
```

[19] ✓ 0.1s

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	60	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	35	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	55	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	56	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...
284	60	1	0	140	207	0	0	138	1	1.9	2	1	3	0
285	46	1	0	140	311	0	1	120	1	1.8	1	2	3	0
286	59	1	3	134	204	0	1	162	0	0.8	2	2	2	0
287	54	1	1	154	232	0	0	164	0	0.0	2	1	2	0
288	53	1	0	110	335	0	1	143	1	3.0	1	1	3	0

289 rows × 14 columns

```
# ambil 13 data dan masukan kedalam variabel data
data = df.iloc[:, :13]
data
```

```
# ambil 13 data dan masukan kedalam variabel data
data = df.iloc[:, :13]
data
```

[21] ✓ 0.1s

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall
0	60	1	3	145	233	1	0	150	0	2.3	0	0	1
1	35	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	55	1	1	120	236	0	1	178	0	0.8	2	0	2
4	56	0	0	120	354	0	1	163	1	0.6	2	0	2
...
284	60	1	0	140	207	0	0	138	1	1.9	2	1	3
285	46	1	0	140	311	0	1	120	1	1.8	1	2	3
286	59	1	3	134	204	0	1	162	0	0.8	2	2	2
287	54	1	1	154	232	0	0	164	0	0.0	2	1	2
288	53	1	0	110	335	0	1	143	1	3.0	1	1	3

289 rows × 13 columns

```
# Rubah data ke array
data = data.values
data
```

```
# Rubah data ke array
data = data.values
data
```

[22] ✓ 0.0s

```
array([[60., 1., 3., ..., 0., 0., 1.],
       [35., 1., 2., ..., 0., 0., 2.],
       [41., 0., 1., ..., 2., 0., 2.],
       ...,
       [59., 1., 3., ..., 2., 2., 2.],
       [54., 1., 1., ..., 2., 1., 2.],
       [53., 1., 0., ..., 1., 1., 3.]])
```

```
# Masukan data pada kolom terakhir dan masukan kedalam
variabel label
label = df.iloc[:, -1]
```

```
▶ # Masukan data pada kolom terakhir dan masukan kedalam variabel label
label = df.iloc[:, -1]
label

[23] ✓ 0.0s

... 0 1
    1 1
    2 1
    3 1
    4 1
    ..
   284 0
   285 0
   286 0
   287 0
   288 0
Name: output, Length: 289, dtype: int64
```

```
# Rubah data ke array
label = label.values
label
```

```
# Bagi data menjadi data training dan data testing
data_train, data_test, label_train, label_test =
train_test_split(data, label, test_size=0.2,
random_state=42)

print('Ukuran data latih:', data_train.shape)
print('Ukuran data uji:', data_test.shape)
```

```
▶ # Bagi data menjadi data training dan data testing
data_train, data_test, label_train, label_test = train_test_split(data, label, test_size=0.2, random_state=42)

print('Ukuran data latih:', data_train.shape)
print('Ukuran data uji:', data_test.shape)

[25] ✓ 0.0s

... Ukuran data latih: (231, 13)
    Ukuran data uji: (58, 13)
```

Cross Validation

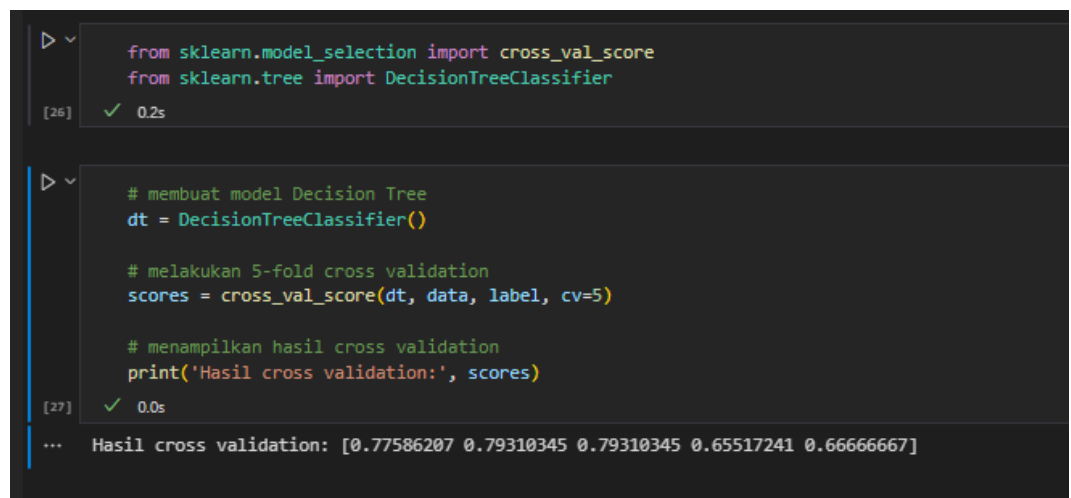
Cross validation merupakan salah satu teknik evaluasi model pada machine learning. Fungsi dari cross validation adalah untuk memperoleh estimasi performa model pada data yang belum pernah dilihat sebelumnya. Dengan melakukan cross validation, kita dapat memperoleh gambaran mengenai kemampuan generalisasi model yang telah dibuat pada dataset yang kita miliki

```
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
```

```
# membuat model Decision Tree
dt = DecisionTreeClassifier()

# melakukan 5-fold cross validation
scores = cross_val_score(dt, data, label, cv=5)

# menampilkan hasil cross validation
print('Hasil cross validation:', scores)
```



```
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier

# membuat model Decision Tree
dt = DecisionTreeClassifier()

# melakukan 5-fold cross validation
scores = cross_val_score(dt, data, label, cv=5)

# menampilkan hasil cross validation
print('Hasil cross validation:', scores)
```

Hasil cross validation: [0.77586207 0.79310345 0.79310345 0.65517241 0.66666667]

model `DecisionTreeClassifier` sebagai classifier dan melakukan 5-fold cross validation pada dataset iris dengan memanggil fungsi `cross_val_score`. Argumen `cv=5` pada fungsi `cross_val_score` menunjukkan bahwa kita akan melakukan 5-fold cross validation.

Jalankan kode program diatas menggunakan Jupyter Notebook atau Google Colab. Kemudian masukan hasil luaran dari kode diatas pada laporan praktikum !

E. TUGAS PRAKTIKUM

1. Implementasikan data preparation menggunakan dataset berikut ini ([iklan_sosmed.csv](#)). Untuk dapat mengerjakan tugas praktikum, anda harus mengerjakan latihan terlebih dahulu. File jupyter notebook sudah tersedia, hanya tinggal melengkapi bagian yang kosong. Silahkan download file jupyter notebook [disini](#).

Data Preparation - Outlier Removal

```
# menghilangkan data point yang memiliki z-score lebih besar dari 3 dan menampilkannya
df_clean = df.loc[round(z_scores) < 3]
# menampilkan data frame yang telah di-clean
print(df_clean)
```

[14] ✓ 0.2s

	ID	Jenis_Kelamin	Umur	Gaji	Transaksi
0	15624510	0	19	285000000	0
1	15810944	0	35	300000000	0
2	15668575	1	26	645000000	0
3	15603246	1	27	855000000	0
4	15804002	0	19	1140000000	0
..
395	15691863	1	46	615000000	1
396	15706071	0	51	345000000	1
397	15654296	1	50	300000000	1
398	15755018	0	36	495000000	0
399	15594041	1	49	540000000	1

[400 rows x 5 columns]

Data Preparation - Mengambil Dataset

```
#Melihat korelasi antar kolom
df_clean.corr()
```

[16] ✓ 0.0s

	ID	Jenis_Kelamin	Umur	Gaji	Transaksi
ID	1.000000	0.025249	-0.000721	0.071097	0.007120
Jenis_Kelamin	0.025249	1.000000	0.073741	0.060435	0.042469
Umur	-0.000721	0.073741	1.000000	0.155238	0.622454
Gaji	0.071097	0.060435	0.155238	1.000000	0.362083
Transaksi	0.007120	0.042469	0.622454	0.362083	1.000000

```
#Mengambil Umur dan Gaji sebagai Variabel Data dan Transaksi sebagai Variabel Label
X = df.iloc[:,2:-1]
y = df.iloc[:, -1]

17] ✓ 0.0s

# Bagi data menjadi 80% data latih dan 20% data uji
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

18] ✓ 0.0s
```

Data Preparation - Normalization

```
# normalisasi data gaji dan umur menggunakan MinMaxScaler
# Inisialisasi MinMaxScaler
scaler = MinMaxScaler()

# Normalisasi data gaji
df['Gaji'] = scaler.fit_transform(df['Gaji'].values.reshape(-1, 1))

# Normalisasi data umur
df['Umur'] = scaler.fit_transform(df['Umur'].values.reshape(-1, 1))

19] ✓ 0.0s
```

Data Preparation - Normalization

```
# Membuat model Decision Tree
model = DecisionTreeClassifier()

# Melakukan 5-fold cross validation
scores = cross_val_score(model, X_train, y_train, cv=5)

# Menampilkan hasil cross validation
print("Hasil Cross Validation (Akurasi):")
print(scores)
print("Rata-rata Akurasi: {:.2f}".format(scores.mean()))

[22] ✓ 0.1s

... Hasil Cross Validation (Akurasi):
[0.859375 0.921875 0.828125 0.875 0.890625]
Rata-rata Akurasi: 0.88
```


F. PENUTUP

1. Kesimpulan

- a. One Hot Encoding mengubah data kategorik dengan membuat kolom baru untuk setiap kategori
- b. Outlier removal adalah teknik penghapusan data yang berada jauh dari sebagian besar data dalam sebuah dataset.
- c. Normalization adalah salah satu teknik yang dipakai dalam data preparation. Tujuan dari normalisasi adalah mengubah nilai-nilai dari sebuah fitur ke dalam skala yang sama. Normalization memungkinkan kenaikan performa dan stabilitas dari sebuah model machine learning.
- d. Standardization adalah proses konversi nilai-nilai dari suatu fitur sehingga nilai-nilai tersebut memiliki skala yang sama.
- e. Dataset yang telah dibersihkan dan diproses kemudian siap kita latih dengan machine learning. Satu-satunya cara untuk mengetahui apakah model machine learning kita bagus atau tidak adalah dengan mengujinya pada kasus atau data baru yang belum dikenali oleh model.
- f. Cross validation merupakan salah satu teknik evaluasi model pada machine learning. Fungsi dari cross validation adalah untuk memperoleh estimasi performa model pada data yang belum pernah dilihat sebelumnya.

2. Saran

Dalam persiapan data, penting untuk melakukan encoding, mengatasi outlier, dan normalisasi/standardisasi. Setelah data siap, latih model dan uji pada data baru. Cross validation dapat memberikan estimasi performa yang baik sebelum menerapkan model pada data yang belum pernah dilihat sebelumnya.