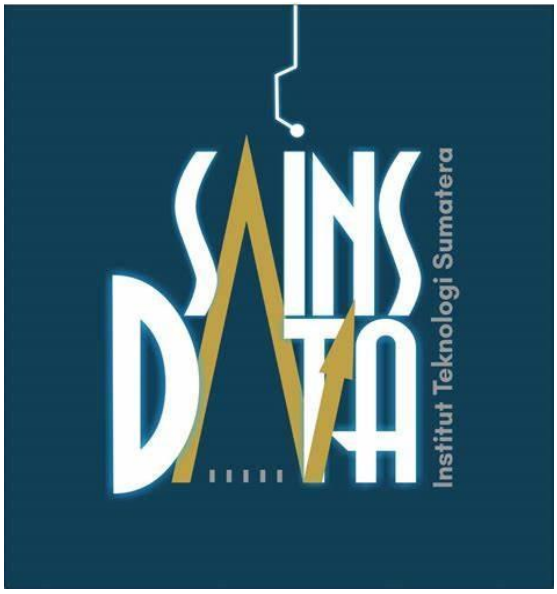
MODUL PRAKTIKUM

MATA KULIAH: DATA MINING

MODUL III

[Data Preprocessing]



Program Studi Sains Data

Fakultas Sains

INSTITUT TEKNOLOGI SUMATERA

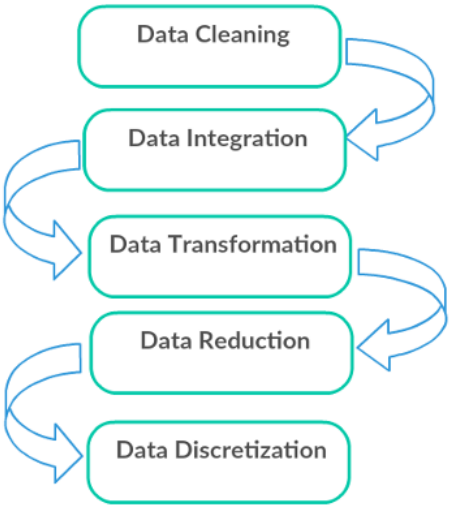
Tahun Ajaran Ganjil 2024/2025.

1. Tujuan Praktikum
    a. Mahasiswa dapat memahami konsep dan praktik terkait dengan Preprocessing data.
    b. Mahasiswa dapat melakukan langkah-langkah menangani noise, missing value dan noise.
    c. Mahasiswa dapat membuat dan memahami terkait dengan analisis korelasi.

2. Tinjauan Pustaka
    a. Konsep Dasar: *Data Preprocessing*

    Data Preprocessing adalah serangkaian langkah atau teknik yang digunakan untuk menyiapkan dan membersihkan data mentah sebelum digunakan dalam analisis lebih lanjut atau dalam model. Process ini dilakukan karena sering kali data mememiliki data yang kurang lengkap atau kosong, terdapat noice, atau format yang tidak sesuai sehingga dapat mempengaruhi keakuratan pada analisi dan model yang dibangun.

    Pengertian lain menyebutkan bahwa data preprocessing adalah tahapan untuk menghilangkan beberapa permasalahan yang bisa mengganggu saat pemrosesan data. Hal tersebut karena banyak data yang formatnya tidak konsisten. Data preprocessing merupakan teknik paling awal sebelum melakukan data mining. Melalui data preprocessing, memungkinkan proses mining akan berjalan dengan lebih efektif dan efisien. Karena data yang telah melalui Pra-pemrosesan data, merupakan data yang sudah melalui beberapa tahap pembersihan. Berikut gamabaran dari tahapan dalam preprocessing data.

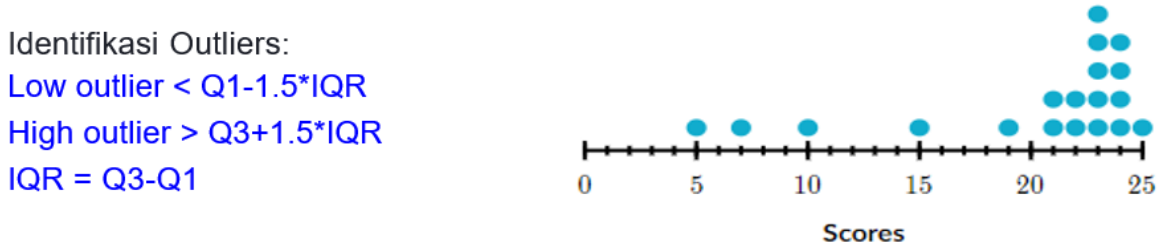


Gambar 1. tahapan data preprocesing data

Tahapan dalam pre-processing adalah:

- **DATA CLEANING**: Tahap pertama yang perlu dilakukan ketika akan preprocessing data adalah data cleaning atau membersihkan data. Artinya, data mentah yang telah diperoleh perlu diseleksi kembali. Kemudian, hapus atau hilangkan data-data yang tidak lengkap, tidak relevan, dan tidak akurat. Dengan melakukan tahap ini, Anda akan menghindari kesalahpahaman ketika menganalisis data tersebut
- **DATA INTEGRATION:** Karena data preprocessing akan menggabungkan beberapa data dalam suatu dataset, maka kita harus mengecek data-data yang datang dari berbagai sumber tersebut supaya memiliki format yang sama.
- **TRANSFORMASI DATA**: Proses berikutnya yang harus dilakukan adalah transformasi data. Seperti yang telah dijelaskan di atas, data akan diambil dari berbagai sumber yang kemungkinan memiliki perbedaan format. Kita harus menyamakan seluruh data yang terkumpul supaya dapat mempermudah proses analisis data
- **MENGURANGI DATA**, Tahap terakhir yang perlu dilakukan adalah mengurangi jumlah data (data reduction). Maksudnya adalah kita harus mengurangi sampel data yang diambil, tetapi dengan catatan, tidak akan mengubah hasil analisis data. Ada tiga teknik yang bisa diterapkan saat melakukan pengurangan data, yakni dimensionality reduction (pengurangan dimensi), numerosity reduction (pengurangan jumlah), dan data compression (kompresi data).

Untuk mengetahui kualitaas data beberapa hal yangperlu diperhatikan yaitu dengan identifikasi nilai yang hilang (missing value), identifikasi kesalahan penentuan tipe data, identifikasi noise atau data yang tidak konsisten, identifikasi pencilan (outlier), dan identifikasi data tidak berimbang (Imbalanced Data).



Gambar 2. Mencari Posisi dan Identifikasi Outlier.

3. Prosedur Praktikum

Dalam praktikumi ini akan dilakukan beberapa tahapan preprocessing dengan *dataset* yang dapat diakses di link https://bit.ly/dataset damin3 dan didalam praktikum memerlukan beberapa *library* pastikan terinstall didalam google colab anda:

- Numpy
- Pandas
- Seaborn
- Scikit-learn

```
pip install scikit-learn
```

Berikut adalah prosedur preprocessing data pada praktikum ini:

a. Menampilkan Data

Data yang digunakan dalam praktikum ini adalah data diabetes_null.csv

```
1 import numpy as np
2 import pandas as pd
3
4 df1 = pd.read_csv('diabetes_null.csv', na_values=['#NAME?'])
5 print(df1.head(5))
```

Akan menampilkan output:

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6    148.0           72.0           35.0      NaN  33.6
1            1     85.0           66.0           29.0      NaN  26.6
2            8    183.0           64.0            NaN      NaN  23.3
3            1     89.0           66.0           23.0     94.0  28.1
4            0    137.0            4.0           35.0    168.0  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627    5        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

Setelah itu kita melihat info dari data kita:

```
1 df1.info()
```

Dengan output :

Note : yang harus kita perhatikan adalah jumlah data kita dan jumlah data disetiap attribute.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   763 non-null    float64
 2   BloodPressure             733 non-null    float64
 3   SkinThickness             541 non-null    float64
 4   Insulin                   394 non-null    float64
 5   BMI                       757 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(6), int64(3)
memory usage: 54.1 KB
```

b. Menangani *Missing Value* (*Handling Missing Values*)

Sebelum menangani missing value kita memerlukan mengidentifikasi *missing value*(nilai yang hilang) yaitu dengan menbuat code :

```
## Identifikasi nilai yang hilang
df1.isnull().sum()
```

Dengan output :

```
Pregnancies                 0
Glucose                     5
BloodPressure              35
SkinThickness             227
Insulin                   374
BMI                        11
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Dari output yang telah kita buat kita dapat mengetahui data yang hilang di setiap baris nya ada berapa dan kita dapat memulai mengatasi data kita, dengan menggunakan beberapa cara seperti dibawah ini:

i. Delete Row :

Salah satu tahap dari preprocessing dimaanfaatkan saat melakukan data cleaning yaitu mempertahankan kualitas data(sehingga data menajdi valid dan lengkap), menghindari bias pada algoritma dan mempermudah proses analisis. Akan tetapi tahap ini harus dipertimbangkan resiko seperti kehilangan informasi dan biasa data dalam sistematis data. oleh karna itu kita harus memahami kapan kita dapat melakukan tahap delete row ini, yaitu saat data kita memiliki proporsi missing value yang rendah dan data yang hilag pada sebuah kolom tidak penting. Contoh tahap delete row dengan menggunkan data diabetes_null.csv:

```
import numpy as np
import pandas as pd

df1 = pd.read_csv('diabetes_null.csv', na_values=['#NAME?'])
df1.isnull().sum().sort_values(ascending=False)
df_no_missing = df1.dropna(axis=0)
print(df_no_missing.head(10))
df_no_missing.info()
```

ii. Fill with mean

salah satu Teknik *handeling missing value* yaitu dengan mengisi nya dengan niali rata-rata (*mean*), fungsi dari fill with mean ini adalah untuk mengurangi bias, mempertahankan jumlah data, meningkatkan kinerja model. Contoh tahapan fill with mean dengan dataset diabetes_null.csv adalah sebagai berikut:

```
import numpy as np
from sklearn.impute import SimpleImputer

# Data from excel
df1 = pd.read_csv('diabetes_null.csv', na_values=['#NAME?'])
imp = SimpleImputer(missing_values=np.nan, strategy='mean')
imp.fit(df1)
df1 = pd.DataFrame(data=imp.transform(df1), columns=df1.columns)
#print
print(df1.head(10))
```

iii. Fill with median

Metode lain untuk menangani missing value adalah mengisinya dengan nilai Tengah(median), tujuan pengisian ini adalah untuk data yang tidak terdistribusi normal dan data yang memiliki outlier atau distribusi yang tidak simetris. Dengan menggunakan library dengan API sklearn.imputer kita dapat melalkukan fill median, berikut contoh dengan menggunakan dataset diabetes_null.csv:

```
from sklearn.impute import SimpleImputer
# Data from excel
df1 = pd.read_csv('diabetes_null.csv', na_values=['#NAME?'])

#Imputer to replace Null with mean
imp = SimpleImputer(missing_values=np.nan, strategy='median')
imp.fit(df1)

df1 = pd.DataFrame(data=imp.transform(df1), columns=df1.columns)
#print
print(df1.head(10))
```

Setelah kita melakukan penanganan missing value yang harus kita perhatikan yaitu terkait dengan jumlah data yang ada di df1 kita serta kita mengecek Kembali apakah dataset kita memiliki data yang hilang atau missing value menggunakan code dibawah ini:

```
df1.isnull().sum()
```

Output:

	0
Pregnancies	0
Glucose	0
BloodPressure	0
SkinThickness	0
Insulin	0
BMI	0
DiabetesPedigreeFunction	0
Age	0
Outcome	0

dtype: int64

Dari gambar output menunjukan disetiap attribute kita sudah tidak ada data yang kosong.

c. Menangani Noise Data (*Handling Noicy Data*)

i. Fungsi untuk menemukaan outlier(pencilan)

Salah satu bentuk dari Noise data adalah Outlier/ pencilan, gambar 2 dapat menggambarkan bagaimana mencari outlier yaitu mencari q1 dan q3 yang masing-masing menggambarkan quartil 1 dan quartil 3 dari data x. setelah mendapatkan q1 dan q3 kita dapat menghitung IQR(*Interquartile Range*), IQR adalah jarak antara kuartil ketiga dan kuartil pertama. IQR digunakan untuk mendeteksi outlier.

```
1 def find_outliers_tukey(x):
2     q1 = x.quantile(.25)
3     q3 = x.quantile(.75)
4     iqr = q3 - q1
5     floor = q1 - 1.5*iqr
6     ceiling = q3 + 1.5*iqr
7     outlier_indices = list(x.index[(x < floor) | (x > ceiling)])
8     outlier_values = list(x[outlier_indices])
9     return outlier_indices, outlier_values
```

Setelah kita mendapatkan nilai IQR kita dapat menentukan nilai floor dan ceiling yaitu:

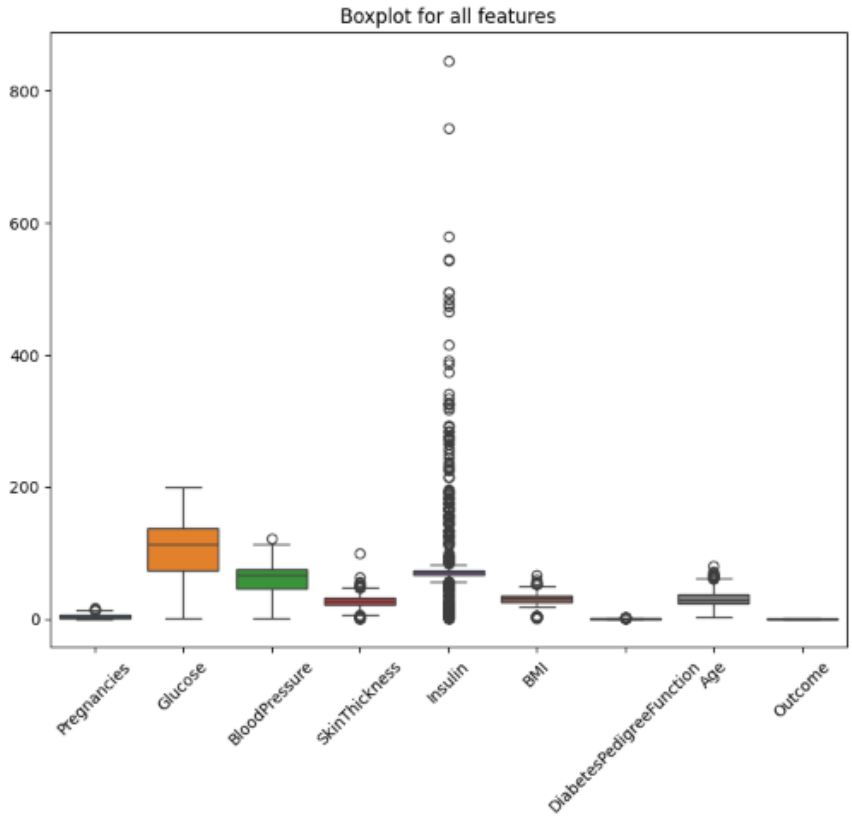
- Floor: nilai batas bawah, yaitu Q1 dikurangi 1.5 kali IQR.
- Ceiling: nilai batas atas, yaitu Q3 ditambah 1.5 kali IQR.

Note: data yang berada dibawah floor dan diatas celling akan dianggap outlier.

Setelah kita mencari nilai dari outlier atau pencilan kita dapat menvisualisasikan outlier disetiap kolom dengan menggunakan code dibawah ini:

```
# membuat plot
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(9, 7))  # Adjust figure size as needed
sns.boxplot(data=df1)
plt.title('Boxplot for all features')
plt.xticks(rotation=45)
plt.show()
```

Output:



Dari gambar boxplot ini kita dapat melihat Gambaran dari setiap atribut apakah ada outlier atau tidak, seperti di variabel insulin terdapat banyak outlier. Untuk menangani outlier dapat dilakukan beberapa hal contohnya seperti Deleting Row(menghapus baris), Menghapus Outlier, Transformasi Data, Winsorizing (Mengganti Outlier dengan Batasan) dan banyak lagi.

ii. Mencari Outlier di Setiap Kolom

Setelah kita membuat fungsi untuk mencari Outlier kita dapat melihat ada beberapa oulier di beberapa attribute sehingga kita perlu mencari nilai daro outlier disetiap kolom nya dengan menggunakan code dibawah ini:

```
glucose_indices, glucose_values = find_outliers_tukey(df1['Glucose'])
print("Outliers for Glucose")
print(np.sort(glucose_values))

print("Outliers for Pregnancies")
pr_indices, pr_values = find_outliers_tukey(df1['Pregnancies'])
print(np.sort(pr_values))

print("Outliers for BloodPressure")
bp_indices, bp_values = find_outliers_tukey(df1['BloodPressure'])
print(np.sort(bp_values))

print("Outliers for SkinThickness")
st_indices, st_values = find_outliers_tukey(df1['SkinThickness'])
print(np.sort(st_values))
```

```
print("Outliers for Insulin")
in_indices, in_values = find_outliers_tukey(df1['Insulin'])
print(np.sort(in_values))

print("Outliers for BMI")
bmi_indices, bmi_values = find_outliers_tukey(df1['BMI'])
print(np.sort(bmi_values))

print("Outliers for DiabetesPedigreeFunction")
dpf_indices, dpf_values = find_outliers_tukey(df1['DiabetesPedigreeFunction'])
print(np.sort(dpf_values))
```

```
print("Outliers for Age")
age_indices, age_values = find_outliers_tukey(df1['Age'])
print(np.sort(age_values))
```

Dalam praktikum modul ini akan mempraktekan beberapa contoh bagaimana cara menangani oulier dengan meng hapus baris (deleting row) dan mengganti dengan nilai mininmal.

- Menangani oulier dengan menghapus baris(deleting row)
  Sebagai contoh untuk menghapus oulier di bp_indices(Outliers for BloodPressure) dengan menggunakan code diabawah ini:

```
df_del = df1.drop(bp_indices)
print(df_del.head(5))
df1.info()
```

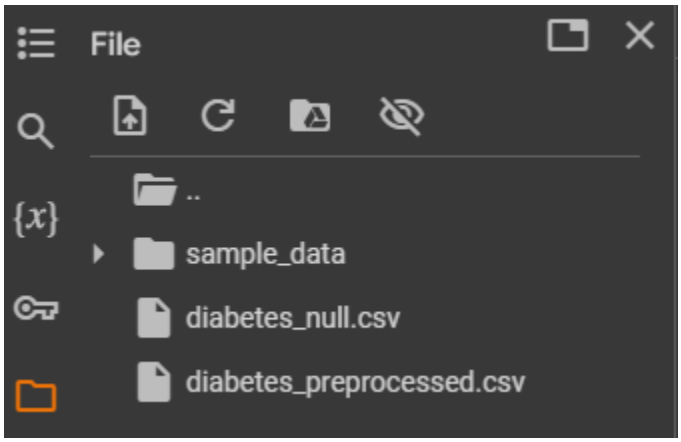
- Menganti dengan nilai min

```
min_in = np.min(df_del['Insulin'])
df_del['Insulin'] = np.where(df_del['Insulin'] > 321, min_in, df_del['Insulin'])
print(df_del.head(10))
```

setelah menghilangkan nilai Outlier dengan kedua metode diatas Langkah selanjutnya yaitu menyimpan data yang telah dihilangkan oulier dengan kode dibawah ini:

```
#buat code menyimpan data ke csv setelah data preprocesing di atas
df_del.to_csv('diabetes_preprocessed.csv', index=False)
df1.info()
```

Download data dengan nama diabetes_preprocessed.csv seperti dibawah ini:



Bandingkan dan amati apa perbedaan antara diabetes_null dan diabetes_preprocessed.csv.

d. Analisis Korelasi (*Correlation Analysis*)

Untuk melihat analisis korelasi kita dapat menggunakan chi-square, **hubungan antara dua variabel kategorikal** dan seberapa jauh distribusi aktual dari data tersebut berbeda dari distribusi yang diharapkan (jika kedua variabel tidak memiliki hubungan). Chi-square test adalah uji statistik yang digunakan untuk menentukan apakah ada hubungan yang signifikan antara dua variabel kategorikal. Berikut formulasi dari chi-square:

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected} \quad (1)$$

Persamaan satu adalah persamaan yang diformulasikan untuk menghitung Chi-Square, dengan X^2 adalah

Chi-square test menghitung nilai berdasarkan perbandingan antara frekuensi yang diamati (observed frequencies) dan frekuensi yang diharapkan

(expected frequencies) jika variabelnya tidak terkait. Salah satu visualisai yang dapat digunakan untuk distribusi frekuensi adalah heat map, heatmap dapat digunakan untuk memvisualisasikan tabel kontingensi. Heatmap memberikan cara yang visual untuk melihat distribusi frekuensi yang diamati dalam dataset. Berikut beberapa visualisasi untuk menggambarkan analisis korelasi:

i. Heatmap

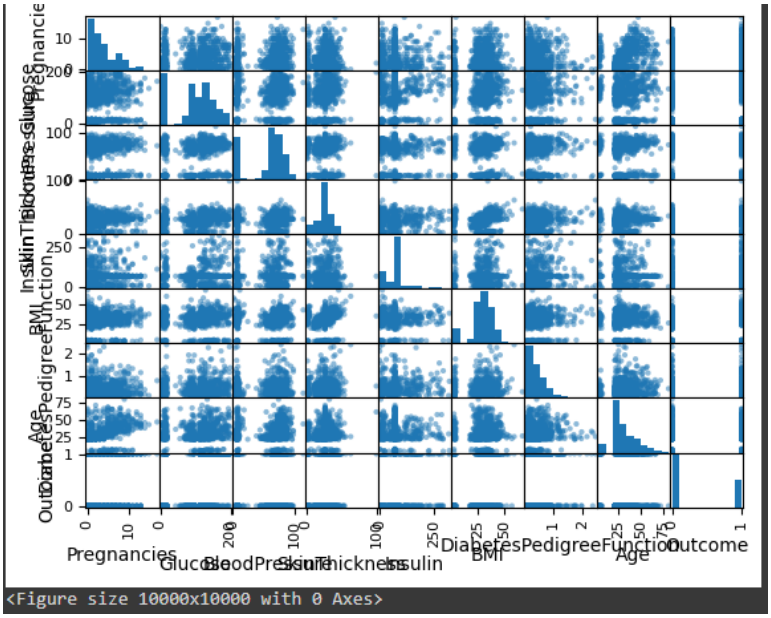
```
import numpy as np
import seaborn as sns
import pandas as pd
from matplotlib import pyplot as plt

df = pd.DataFrame(df_del)
df= df.iloc[:].copy()
corr = df.corr()
plt.figure(figsize=(11,8))
sns.heatmap(corr, cmap="Greens",annot=True)
plt.show()
```

Output dari heat map diatas adalah sebagai berikut:

ii. Visually Evaluating Correlation

```
from pandas.plotting import scatter_matrix
scatter_matrix(df_del)
plt.figure(figsize=(100, 100))
plt.show()
```

Output dari scatter plot adalah sebagai berikut:



Note:Scatter plots showing the similarity from –1 to 1.