LAPORAN PEMBELAJARAN MESIN

KELAS D

“Data Preparation”

DOSEN PROGRAM STUDI

Dr. Ir. Ridowati Gunawan S.kom., M.T.

Disusun Oleh :

Victoria Alysha Fernando S

215314158

PROGRAM STUDI INFORMATIKA

FAKULTAS SAINS DAN TEKNOLOGI

UNIVERSITAS SANATA DHARMA

YOGYAKARTA

2023

1. Analisis terhadap data yang kosong

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
df=pd.read_csv("mesin_data_sks_ipk.csv")
df.head()
```

Out[2]:

	Nama	SKS	IPK	ANGKATAN
0	Barcelius Barapadang	81	3.47	20
1	Ardian Vega Carrelino	84	3.64	21
2	Rosalin Gaudiensia br Ginting	84	3.98	21
3	Resiana Kinanti Jati	81	3.80	21
4	Loadtriani Oktavia S	87	3.77	21

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Nama      32 non-null     object
 1   SKS       32 non-null     int64
 2   IPK       31 non-null     float64
 3   ANGKATAN  32 non-null     int64
dtypes: float64(1), int64(2), object(1)
memory usage: 1.1+ KB
```

In [4]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [5]:
```
df['Nama']=le.fit_transform(df['Nama'])
```

In [6]:
```
df.head()
```

Out[6]:

	Nama	SKS	IPK	ANGKATAN
0	5	81	3.47	20
1	1	84	3.64	21
2	24	84	3.98	21
3	21	81	3.80	21
4	16	87	3.77	21

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Nama      32 non-null     int32
 1   SKS       32 non-null     int64
 2   IPK       31 non-null     float64
 3   ANGKATAN  32 non-null     int64
dtypes: float64(1), int32(1), int64(2)
memory usage: 1.0 KB
```

In [8]:
```
y=df['ANGKATAN']
```

In [9]:
```
df.drop('ANGKATAN', axis=1, inplace=True)
```

In [10]:
```
df.info()
```

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Nama    32 non-null     int32
 1   SKS     32 non-null     int64
 2   IPK     31 non-null     float64
dtypes: float64(1), int32(1), int64(1)
memory usage: 768.0 bytes
```

In [11]:
```
updated_df = df.dropna(axis=1)
```

In [12]:
```
updated_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Nama    32 non-null     int32
 1   SKS     32 non-null     int64
dtypes: int32(1), int64(1)
memory usage: 512.0 bytes
```

In [13]:
```
# pembagian data training testing
from sklearn import metrics
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(updated_df,y,test_size=0.3)
#membangun model
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(X_train,y_train)
pred=lr.predict(X_test)
print(metrics.accuracy_score(pred,y_test))
```

```
0.9
```

In [14]:
```
newdf=pd.read_csv("mesin_data_sks_ipk.csv")
newdf.head()
```

Out[14]:

	Nama	SKS	IPK	ANGKATAN
0	Barcelius Barapadang	81	3.47	20
1	Ardian Vega Carrelino	84	3.64	21
2	Rosalin Gaudiensia br Ginting	84	3.98	21
3	Resiana Kinanti Jati	81	3.80	21
4	Loadtriani Oktavia S	87	3.77	21

In [15]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
newdf['Nama']=le.fit_transform(newdf['Nama'])
```

In [16]:
```
newdf.info()
```

In [16]:
```
newdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Nama      32 non-null     int32
 1   SKS       32 non-null     int64
 2   IPK       31 non-null     float64
 3   ANGKATAN  32 non-null     int64
dtypes: float64(1), int32(1), int64(2)
memory usage: 1.0 KB
```

In [17]:
```
updated_df=newdf.dropna(axis=0)
```

In [18]:
```
updated_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 31 entries, 0 to 31
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Nama      31 non-null     int32
 1   SKS       31 non-null     int64
 2   IPK       31 non-null     float64
 3   ANGKATAN  31 non-null     int64
dtypes: float64(1), int32(1), int64(2)
memory usage: 1.1 KB
```

In [19]:
```
y1=updated_df['ANGKATAN']
updated_df.drop('ANGKATAN',axis=1,inplace=True)
```

```
C:\Users\sasha\AppData\Local\Temp\ipykernel_15732\3813776026.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  updated_df.drop('ANGKATAN',axis=1,inplace=True)
```

In [20]:
```
updated_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 31 entries, 0 to 31
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Nama    31 non-null     int32
 1   SKS     31 non-null     int64
 2   IPK     31 non-null     float64
dtypes: float64(1), int32(1), int64(1)
memory usage: 868.0 bytes
```

In [21]:
```
# pembagian data training testing
from sklearn import metrics
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(updated_df,y1,test_size=0.3)
#membangun model
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(X_train,y_train)
pred=lr.predict(X_test)
print(metrics.accuracy_score(pred,y_test))
```

```
1.0
```

In [22]:
```
updated_df_fill=df
```

In [23]:
```
updated_df_fill.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Nama    32 non-null     int32
 1   SKS     32 non-null     int64
 2   IPK     31 non-null     float64
dtypes: float64(1), int32(1), int64(1)
memory usage: 768.0 bytes
```

In [24]:
```
updated_df_fill['IPK']=updated_df_fill['IPK'].fillna(updated_df_fill['IPK'].mean())
```

In [25]:
```
updated_df_fill.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Nama    32 non-null     int32
 1   SKS     32 non-null     int64
 2   IPK     32 non-null     float64
dtypes: float64(1), int32(1), int64(1)
memory usage: 768.0 bytes
```

In [26]:
```
from sklearn import metrics
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(updated_df_fill,y,test_size=0.3)
#membangun model
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(X_train,y_train)
pred=lr.predict(X_test)
print(metrics.accuracy_score(pred,y_test))
```

```
1.0
```

In [27]:
```
updated_df_si = df
updated_df_si['IPKismissing']=updated_df_si['IPK'].isnull()
from sklearn.impute import SimpleImputer
my_imputer=SimpleImputer(strategy='median')
data_new=my_imputer.fit_transform(updated_df_si)
```

In [28]:
```
updated_df_si.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32 entries, 0 to 31
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Nama          32 non-null     int32
 1   SKS           32 non-null     int64
 2   IPK           32 non-null     float64
 3   IPKismissing  32 non-null     bool
dtypes: bool(1), float64(1), int32(1), int64(1)
memory usage: 800.0 bytes
```

2. Lakukan normalisasi minmax dan z-score untuk data SKS dan IPK

Normalisasi

In [31]:
```
newdf = df[['IPK', 'SKS']]
newdf
```

Out[31]:

	IPK	SKS
0	3.470000	81
1	3.640000	84
2	3.980000	84
3	3.800000	81
4	3.770000	87
5	3.880000	86
6	3.730000	81
7	4.900000	155
8	3.880000	81
9	2.750000	80
10	3.440000	131
11	3.320000	128
12	2.960000	20
13	3.700000	86
14	3.910000	81
15	3.680000	81
16	2.290000	22
17	3.720000	19
18	3.800000	88
19	3.110000	21
20	3.480323	85
21	2.850000	82
22	3.840000	81
23	3.120000	77
24	3.100000	85
25	3.500000	131
26	3.000000	80
27	3.780000	18
28	3.100000	80
29	3.100000	21
30	3.400000	24
31	3.370000	21

In [32]:
```
from sklearn.preprocessing import MinMaxScaler, StandardScaler

zscore = StandardScaler()
minmax = MinMaxScaler()

scaler = zscore.fit(newdf)
scaler_m = minmax.fit(newdf)
```

In [33]:
```
df_z = scaler.transform(newdf)
df_z = pd.DataFrame(df_z)
df_z
```

Out[33]:

	0	1
0	-0.021905	0.202533
1	0.338837	0.287068
2	1.060320	0.287068
3	0.678358	0.202533
4	0.614698	0.371604
5	0.848119	0.343425
6	0.529818	0.202533
7	3.012568	2.287741
8	0.848119	0.202533
9	-1.549751	0.174354
10	-0.085565	1.611458
11	-0.340206	1.526922
12	-1.104129	-1.516355
13	0.466157	0.343425
14	0.911779	0.202533
15	0.423717	0.202533
16	-2.525875	-1.459998
17	0.508598	-1.544534
18	0.678358	0.399782
19	-0.785828	-1.488177
20	0.000000	0.315247
21	-1.337550	0.230711
22	0.763239	0.202533
23	-0.764608	0.089819
24	-0.807048	0.315247
25	0.041756	1.611458
26	-1.019249	0.174354
27	0.635918	-1.572712
28	-0.807048	0.174354
29	-0.807048	-1.488177
30	-0.170445	-1.403641
31	-0.234106	-1.488177

In [34]:
```
df_m = scaler_m.transform(newdf)
df_m = pd.DataFrame(df_m)
df_m
```

Out[34]:

	0	1
0	0.452107	0.459854
1	0.517241	0.481752
2	0.647510	0.481752
3	0.578544	0.459854
4	0.567050	0.503650
5	0.609195	0.496350
6	0.551724	0.459854
7	1.000000	1.000000
8	0.609195	0.459854
9	0.176245	0.452555
10	0.440613	0.824818
11	0.394636	0.802920
12	0.256705	0.014599
13	0.540230	0.496350
14	0.620690	0.459854
15	0.532567	0.459854
16	0.000000	0.029197
17	0.547893	0.007299
18	0.578544	0.510949
19	0.314176	0.021898
20	0.456062	0.489051
21	0.214559	0.467153
22	0.593870	0.459854
23	0.318008	0.430657
24	0.310345	0.489051
25	0.463602	0.824818
26	0.272031	0.452555
27	0.570881	0.000000
28	0.310345	0.452555
29	0.310345	0.021898
30	0.425287	0.043796
31	0.413793	0.021898

MinMax dan Z-Score

In [35]:
```
new_normal = df.copy()
new_normal['Z-SKS'] = df_z[0]
new_normal['Z-IPK'] = df_z[1]
new_normal['MinMax-SKS'] = df_m[0]
new_normal['MinMax-IPK'] = df_m[1]
new_normal
```

Out[35]:

	Nama	SKS	IPK	IPKismissing	Z-SKS	Z-IPK	MinMax-SKS	MinMax-IPK
0	5	81	3.470000	False	-0.021905	0.202533	0.452107	0.459854
1	1	84	3.640000	False	0.338837	0.287068	0.517241	0.481752
2	24	84	3.980000	False	1.060320	0.287068	0.647510	0.481752
3	21	81	3.800000	False	0.678358	0.202533	0.578544	0.459854
4	16	87	3.770000	False	0.614698	0.371604	0.567050	0.503650
5	23	86	3.880000	False	0.848119	0.343425	0.609195	0.496350
6	15	81	3.730000	False	0.529818	0.202533	0.551724	0.459854
7	26	155	4.900000	False	3.012568	2.287741	1.000000	1.000000
8	18	81	3.880000	False	0.848119	0.202533	0.609195	0.459854
9	12	80	2.750000	False	-1.549751	0.174354	0.176245	0.452555
10	9	131	3.440000	False	-0.085565	1.611458	0.440613	0.824818
11	17	128	3.320000	False	-0.340206	1.526922	0.394636	0.802920
12	25	20	2.960000	False	-1.104129	-1.516355	0.256705	0.014599
13	27	86	3.700000	False	0.466157	0.343425	0.540230	0.496350
14	28	81	3.910000	False	0.911779	0.202533	0.620690	0.459854
15	0	81	3.680000	False	0.423717	0.202533	0.532567	0.459854
16	19	22	2.290000	False	-2.525875	-1.459998	0.000000	0.029197
17	6	19	3.720000	False	0.508598	-1.544534	0.547893	0.007299
18	20	88	3.800000	False	0.678358	0.399782	0.578544	0.510949
19	14	21	3.110000	False	-0.785828	-1.488177	0.314176	0.021898
20	2	85	3.480323	False	0.000000	0.315247	0.456062	0.489051
21	22	82	2.850000	False	-1.337550	0.230711	0.214559	0.467153
22	11	81	3.840000	False	0.763239	0.202533	0.593870	0.459854
23	13	77	3.120000	False	-0.764608	0.089819	0.318008	0.430657
24	8	85	3.100000	False	-0.807048	0.315247	0.310345	0.489051
25	7	131	3.500000	False	0.041756	1.611458	0.463602	0.824818
26	4	80	3.000000	False	-1.019249	0.174354	0.272031	0.452555
27	30	18	3.780000	False	0.635918	-1.572712	0.570881	0.000000
28	10	80	3.100000	False	-0.807048	0.174354	0.310345	0.452555
29	29	21	3.100000	False	-0.807048	-1.488177	0.310345	0.021898
30	31	24	3.400000	False	-0.170445	-1.403641	0.425287	0.043796
31	3	21	3.370000	False	-0.234106	-1.488177	0.413793	0.021898

In [36]:
```
new_normal_z = new_normal.copy()
new_normal_z = new_normal_z.drop(columns=['SKS', 'IPK', 'IPKismissing', 'MinMax-SKS', 'MinMax-IPK'], axis=1)
new_normal_z
```

Out[36]:

	Nama	Z-SKS	Z-IPK
0	5	-0.021905	0.202533
1	1	0.338837	0.287068
2	24	1.060320	0.287068
3	21	0.678358	0.202533
4	16	0.614698	0.371604
5	23	0.848119	0.343425
6	15	0.529818	0.202533
7	26	3.012568	2.287741
8	18	0.848119	0.202533
9	12	-1.549751	0.174354
10	9	-0.085565	1.611458
11	17	-0.340206	1.526922
12	25	-1.104129	-1.516355
13	27	0.466157	0.343425
14	28	0.911779	0.202533
15	0	0.423717	0.202533
16	19	-2.525875	-1.459998
17	6	0.508598	-1.544534
18	20	0.678358	0.399782
19	14	-0.785828	-1.488177
20	2	0.000000	0.315247
21	22	-1.337550	0.230711
22	11	0.763239	0.202533
23	13	-0.764608	0.089819
24	8	-0.807048	0.315247
25	7	0.041756	1.611458
26	4	-1.019249	0.174354
27	30	0.635918	-1.572712
28	10	-0.807048	0.174354
29	29	-0.807048	-1.488177
30	31	-0.170445	-1.403641
31	3	-0.234106	-1.488177

In [37]:
```
from sklearn import metrics
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(new_normal_z,y,test_size=0.3)
#membangun model
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(X_train,y_train)
pred=lr.predict(X_test)
print(metrics.accuracy_score(pred,y_test))
```

```
0.9
```

3. Perlihatkan pengaruh dari normalisasi dan penanganan missing value

Pengaruh dari normalisasi dan penanganan *missing value* yaitu adanya perubahan ke-akurasi-an data, dapat dilihat dari tabel berikut :

Metode	Akurasi
Dropna(axis=1)	0.9
Dropna(axis=0)	1.0
.fillna(.mean())	1.0
SimpleImputer(strategy='median')	1.0
MinMax	0.9

Dari data tabel diatas, hasil akurasi berubah-ubah akibat pengaruh dari normalisasi dan penanganan *missing value* dengan hasil angka yang mendekati sempurna (0.9) atau sempurna (1.0).

Berikut link Gform: https://forms.gle/v2eGuMZUZfjoJoQa9