**Vidyavardhini's College of Engineering & Technology**
Department of Computer Engineering
Academic Year: 2025-26

| |
|---|
| Experiment No.3 |
| Perform data pre-processing |
| Name: Sumit Metkari |
| Div: TE-2 |
| Roll no: 32 |
| Date of Performance: 30-07-25 |
| Date of Submission: 06-08-25 |

**Aim:** To implement data preprocessing Algorithm

**Objective:-**Develop a program to implement data preprocessing algorithm

**Theory:**Why preprocess the data? Because data in the real world is dirty, incomplete and noisy. Incomplete in lacking attributes values and lacking attributes of interest or containing only aggregate value noisy in terms of containing errors or outliers and inconsistent containing discrepancies in names or codes. Now the question arises why is the data dirty? Because incomplete data may come from ―not applicable‖ data value when data has to be collected and the major issue is a different consideration between the times when the data was analyzed and human hardware and software issues are common. Noisy data may come from the when a human enters the wrong value at the time of data entry as Nobody is perfect. Errors in transmission of data and instruments that collect the faulty data. Inconsistent data may come from the different data sources. Duplicates records also need data cleaning.

Why data preprocessing is important? Data is not clean, Duplicity of data and the no quality data and the most important is no quality result so data preprocessing is important. Quality decisions must be based on the quality data. Data warehouse needs consistent integration of quality data. By the processing of data, data quality can be measures in term of accuracy, completeness, consistency, timeliness, believability, interpretability. There are three methods to handle the noisy data.

The different pre-processing steps that can be applied are:

1) Filling up the missing values
2) Removing duplicate data
3) Handling noisy data
4) Handling outliers
5) Scaling of data
6) Encoding of text or categorical values

**Code and output:**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
dataset = pd.read_csv("C:/Users/DELL/Downloads/diabetes - diabetes.csv"
")
```

```
x= dataset.iloc[:,:-1].values
y= dataset.iloc[:,-1].values
print(y)
print(x)
print(dataset)
```

```
[1 0 1 0 1 0 1 0 1 1 0 1 0 1 1 1 1 1 0 1 0 0 1 1 1 1 1 0 0 0 0 1 0 0 0 0 0
 1 1 1 0 0 0 1 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 1 0 1 0
 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 1
 1 0 0 1 1 1 0 0 0 1 0 0 0 1 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
 0 0 0 0 1 0 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 1 0 1 0 1 0 0 0 0 0
 1 1 1 1 1 0 0 1 1 0 1 0 1 1 1 0 0 0 0 0 1 1 0 1 0 0 0 1 1 1 1 0 1 1 1 1
 0 0 0 0 0 1 0 0 1 1 0 0 0 1 1 1 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0
 1 0 1 0 0 1 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 1 1 0 0
 1 0 1 0 1 1 0 1 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0 1 0 1 1 1 0 0 1 0 1 0 0 0 1
 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 1 1 0 0 1 0 0 1 0 0 1
 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 1 0 0 1 0 1 1 0 1 0 1 0 1
 0 1 1 0 0 0 0 1 1 0 1 0 1 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1
 1 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1
 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 1 1 0
 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0
 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 1 0 1 0 1 0 1 0
 1 0 0 1 0 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 1 0
 1 1 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0 1 0 0 0 0 0 1 0 1 0 1 0 1 1 0 0 0 0 1 1
 0 0 0 1 0 1 1 0 0 1 0 0 1 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1
 1 0 0 1 0 0 1 0 1 1 1 0 0 1 1 1 0 1 0 1 0 1 0 0 0 0 1 0]
[[  6.    148.     72.    ...  33.6     0.627  50.   ]
 [  1.     85.     66.    ...  26.6     0.351  31.   ]
 [  8.    183.     64.    ...  23.3     0.672  32.   ]
 ...
 [  5.    121.     72.    ...  26.2     0.245  30.   ]
 [  1.    126.     60.    ...  30.1     0.349  47.   ]
 [  1.     93.     70.    ...  30.4     0.315  23.   ]]
```

```
...      Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
    0              6      148             72             35        0  33.6
    1              1       85             66             29        0  26.6
    2              8      183             64              0        0  23.3
    3              1       89             66             23       94  28.1
    4              0      137             40             35      168  43.1
    ..           ...      ...            ...            ...      ...   ...
    763           10      101             76             48      180  32.9
    764            2      122             70             27        0  36.8
    765            5      121             72             23      112  26.2
    766            1      126             60              0        0  30.1
    767            1       93             70             31        0  30.4

         DiabetesPedigreeFunction  Age  Outcome
    0                       0.627   50        1
    1                       0.351   31        0
    2                       0.672   32        1
    3                       0.167   21        0
    4                       2.288   33        1
    ..                        ...  ...      ...
    763                     0.171   63        0
    764                     0.340   27        0
    765                     0.245   30        0
    766                     0.349   47        1
    767                     0.315   23        0

    [768 rows x 9 columns]
```

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(x[:,1:3])
x[:,1:3]=imputer.transform(x[:,1:3])
print(x)
```

```
[[  6.    148.     72.    ...  33.6     0.627  50.   ]
 [  1.     85.     66.    ...  26.6     0.351  31.   ]
 [  8.    183.     64.    ...  23.3     0.672  32.   ]
 ...
 [  5.    121.     72.    ...  26.2     0.245  30.   ]
 [  1.    126.     60.    ...  30.1     0.349  47.   ]
 [  1.     93.     70.    ...  30.4     0.315  23.   ]]
```

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='passthrough')
x = np.array(ct.fit_transform(x))
print(x)
```

```
[[ 0.     0.     0.    ... 33.6    0.627 50.   ]
 [ 0.     1.     0.    ... 26.6    0.351 31.   ]
 [ 0.     0.     0.    ... 23.3    0.672 32.   ]
 ...
 [ 0.     0.     0.    ... 26.2    0.245 30.   ]
 [ 0.     1.     0.    ... 30.1    0.349 47.   ]
 [ 0.     1.     0.    ... 30.4    0.315 23.   ]]
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = np.array(le.fit_transform(y))
print(y)
```

```
[1 0 1 0 1 0 1 0 1 1 0 1 0 1 1 1 1 1 0 1 0 0 1 1 1 1 1 0 0 0 0 1 0 0 0 0 0
 1 1 1 0 0 0 1 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 1 0 1 0
 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 1
 1 0 0 1 1 1 0 0 0 1 0 0 0 1 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
 0 0 0 0 1 0 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 1 0 1 0 1 0 0 0 0 0
 1 1 1 1 1 0 0 1 1 0 1 0 1 1 1 0 0 0 0 0 0 1 1 0 1 0 0 0 1 1 1 1 0 1 1 1 1
 0 0 0 0 0 1 0 0 1 1 0 0 0 1 1 1 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0
 1 0 1 0 0 1 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 1 1 0 0
 1 0 1 0 1 1 0 1 0 0 1 0 1 1 0 0 1 0 1 0 0 1 0 1 0 1 1 1 0 0 1 0 1 0 0 0 1
 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 1 1 0 0 1 0 0 1 0 0 1
 1 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 1 0 0 1 0 1 1 0 1 0 1 0 1
 0 1 1 0 0 0 0 1 1 0 1 0 1 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1
 1 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1
 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 1 0
 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0
 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 1 0 1 0 1 0 1 0
 1 0 0 1 0 0 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 1 0 0 0 1 0 1 1 1 1 0
 1 1 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0 1 0 0 0 0 0 1 0 1 0 1 0 1 1 0 0 0 0 1 1
 0 0 0 1 0 1 1 0 0 1 0 0 1 1 0 0 1 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1
 1 0 0 1 0 0 1 0 1 1 1 0 0 1 1 1 0 1 0 1 0 1 0 0 0 0 1 0]
```

print(dataset['Age'].describe())

```
count    768.000000
mean      33.240885
std       11.760232
min       21.000000
25%       24.000000
50%       29.000000
75%       41.000000
max       81.000000
Name: Age, dtype: float64
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 1)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train[:, 3:] = sc.fit_transform(x_train[:, 3:])
x_test[:, 3:] = sc.transform(x_test[:, 3:])
print(x_test)
```

```
[[ 0.          0.          0.         ... -0.76497935  0.56009786
   1.50008581]
 [ 0.          1.          0.         ... -0.75186334 -0.87067912
  -0.95741055]
 [ 0.          0.          0.         ... -0.89613942 -0.78813429
  -0.53370428]
 ...
 [ 0.          0.          0.         ...  2.10742614 -0.99908218
   0.82215578]
 [ 0.          0.          0.         ... -4.17514112  0.53869735
   3.02542839]
 [ 0.          0.          0.         ... -0.50265921  0.44698088
  -0.19473927]]
```

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(x_train, y_train)
x_predict = np.array(x_test[:,:])
print(classifier.predict((x_predict)))
accuracy = classifier.score(x_test,y_test)
print(accuracy)
```

```
[1 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 1 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 1 0
 1 0 0 0 0 0 1 0 0 1 0 0 0 0 1 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 1 1 0 1 1 0 0
 0 1 1 1 0 1 1 0 0 0 0 0 1 1 1 0 0 0 1 0 1 1 0 0 1 0 0 0 1 0 0 0 0 1 0 0 1
 0 0 1 0 0 1 0 1 1 0 1 0 0 0 0 1 1 0 0 1 0 1 1 0 0 0 1 0 0 1 0 0 1 0 1 0 0
 0 1 0 1 0 1]
0.6753246753246753
```

```
print(y_test)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 1 1 1 1 0 0 0 1 0 1 1 0 0 1 0 1 0
 0 0 0 0 0 0 1 0 0 1 1 0 1 0 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 1 1 0 1 1 0 0 0
 1 1 1 0 0 1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 1 0 1
 0 0 0 1 0 0 1 0 1 0 1 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0 1 0 0 1 1 1 0 0
 1 0 0 1 0 0]
```

```
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(x_train, y_train)
```

**Conclusion:**

Data pre-processing is a crucial step in data analysis and machine learning because raw data is often incomplete, inconsistent, and noisy. Pre-processing ensures data quality by handling missing values, removing outliers, encoding categorical variables, and scaling features so that they are comparable. It improves model accuracy, speeds up computation, and prevents biases caused by irregular or unbalanced data.

Without pre-processing, algorithms may fail to run, produce inaccurate or misleading results, take longer to train, or learn patterns that are irrelevant or incorrect. In short, the quality of the model's output depends directly on the quality of the data fed into it.