



Vidyavardhini's College of Engineering and Technology

Department of Computer Engineering

Academic Year : 2023-24 (Odd Sem)

Experiment No.3
Perform data pre-processing
Date of Performance:
Date of Submission:



Vidyavardhini's College of Engineering and Technology

Department of Computer Engineering

Academic Year : 2023-24 (Odd Sem)

**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:



Vidyavardhini's College of Engineering and Technology

Department of Computer Engineering

Academic Year : 2023-24 (Odd Sem)

A screenshot of a Google Colab notebook titled "Copy of Data pre processing.ipynb". The notebook contains Python code for data preprocessing. The code includes imports for pandas, numpy, and matplotlib, reading a CSV file named "Data - data.csv", and displaying the first few rows of the dataset. The output shows a list of rows with columns: Country, Age, Salary, Purchased, and No. The data includes entries for France, Spain, Germany, and the UK, with some missing values (NaN) for Age and Salary. The notebook interface shows the code editor, output area, and file explorer. The bottom status bar indicates the notebook is completed at 10:56 AM on 01-09-2023.

```
[1] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

[4] dataset = pd.read_csv('Data - data.csv')

[5] x= dataset.iloc[:,1:3].values
y= dataset.iloc[:,3].values
print(y)
print(x)

[["No", "yes", "No", "No", "yes", "yes", "No", "yes", "No", "yes"]
[["France", 44.0, 72000.0],
["Spain", 27.0, 46000.0],
["Germany", 30.0, 53000.0],
["Spain", 38.0, 61000.0],
["Germany", 40.0, nan],
["France", 25.0, 58000.0],
["Spain", nan, 52000.0],
["France", 40.0, 79000.0],
["Germany", 50.0, 83000.0],
["France", 37.0, 67000.0]]

[6] print(dataset)
```

Country	Age	Salary	Purchased	No
France	44.0	72000.0		No

```

[6]: print(DataSet)

Country  Age  Salary  Purchased
0  France  44.0  72000.0      No
1  Spain   27.0  48000.0      Yes
2  Germany 30.0  53000.0      No
3  Spain   36.0  61000.0      No
4  Germany 40.0    840.0      Yes
5  France  25.0  58000.0      Yes
6  Spain   36.0  52000.0      No
7  France  40.0  79000.0      Yes
8  Germany 50.0  83000.0      No
9  France  37.0  67000.0      Yes

[7]: 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])

[8]: print(X)

[["France" 44.0 72000.0]
 ["Spain" 27.0 48000.0]
 ["Germany" 30.0 53000.0]
 ["Spain" 36.0 61000.0]
 ["Germany" 40.0 51777.77777777778]
 ["France" 25.0 58000.0]
 ["Spain" 36.77777777777778 52000.0]
 ["France" 40.0 79000.0]
 ["Germany" 50.0 83000.0]
 ["France" 37.0 67000.0]]

```



Vidyavardhini's College of Engineering and Technology

Department of Computer Engineering

Academic Year : 2023-24 (Odd Sem)

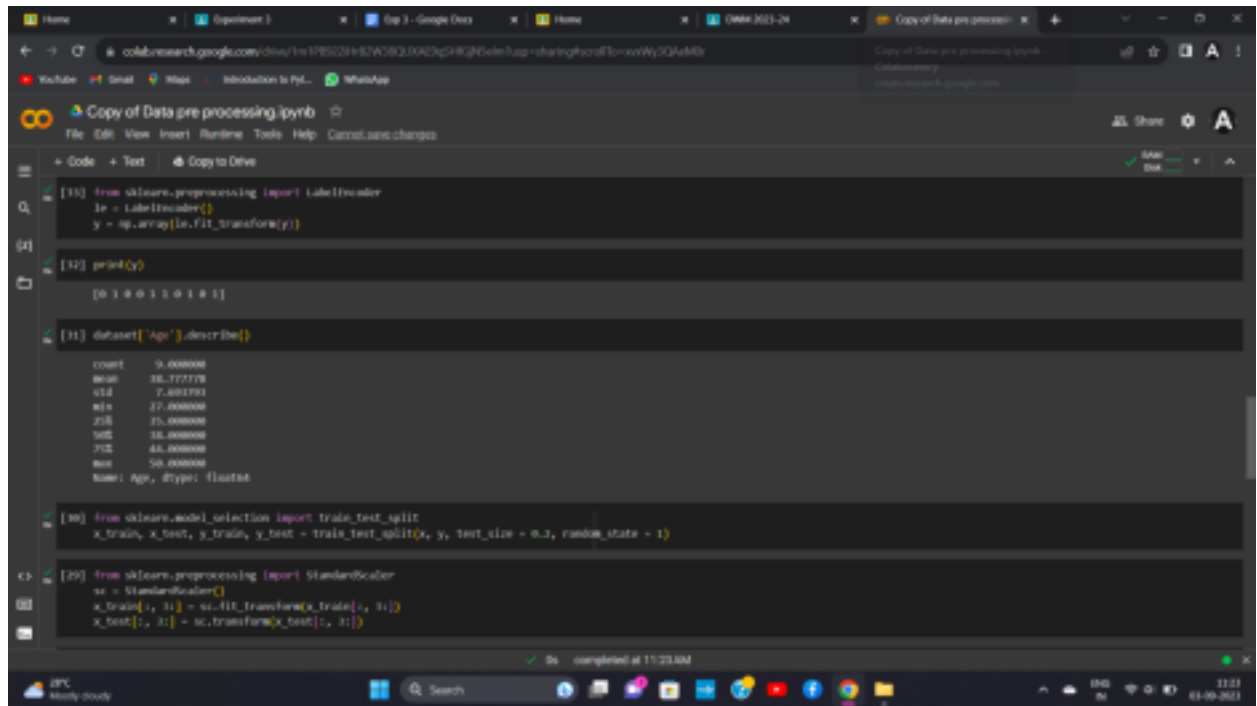
```

[9]: 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))

[10]: print(X)

[[1.0 0.0 0.0 44.0 72000.0]
 [0.0 0.0 1.0 27.0 48000.0]
 [0.0 1.0 0.0 30.0 53000.0]
 [0.0 0.0 1.0 36.0 61000.0]
 [0.0 1.0 0.0 40.0 51777.77777777778]
 [1.0 0.0 0.0 25.0 58000.0]
 [0.0 0.0 1.0 36.77777777777778 52000.0]
 [1.0 0.0 0.0 40.0 79000.0]
 [0.0 1.0 0.0 50.0 83000.0]
 [1.0 0.0 0.0 37.0 67000.0]]

```



```
[11] from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = np.array(le.fit_transform(y))

[12] print(y)

[0 1 0 0 1 0 1 0 1]

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

count    9.000000
mean     38.777778
std       7.401793
min       27.000000
25%       35.000000
50%       38.000000
75%       46.000000
max       50.000000
Name: Age, dtype: float64

[10] from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 1)

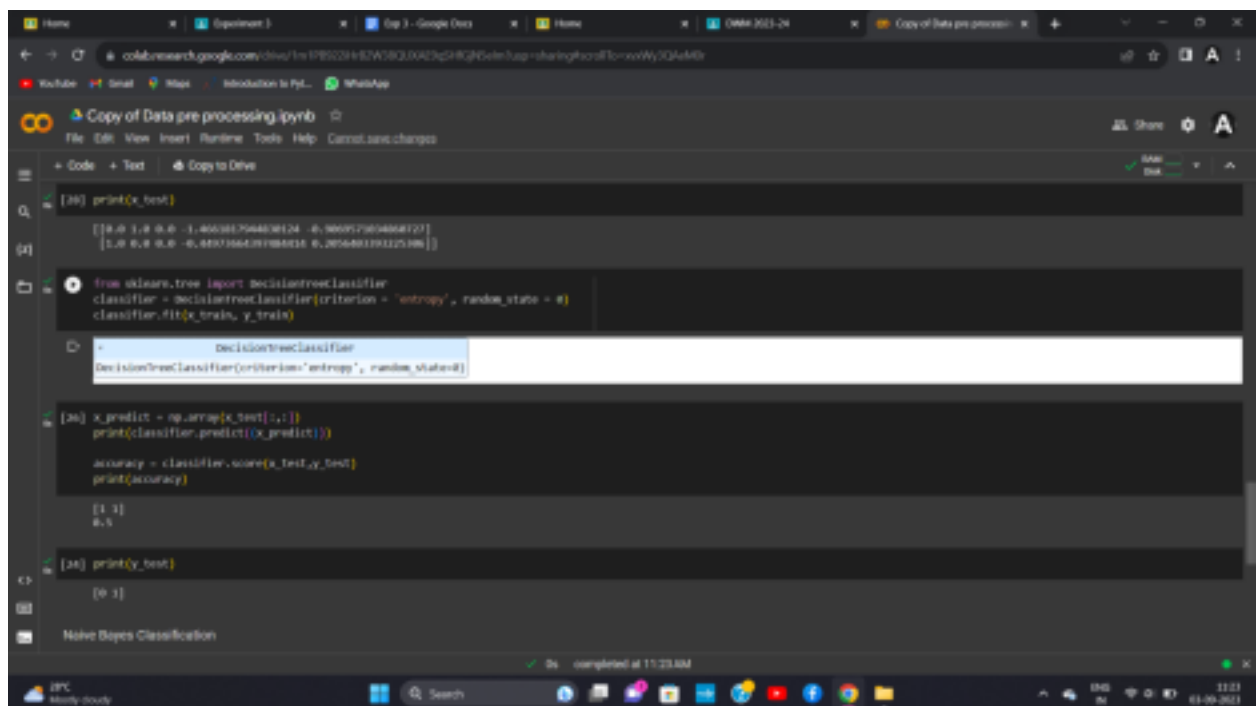
[29] from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train[:, 1:] = sc.fit_transform(x_train[:, 1:])
x_test[:, 1:] = sc.transform(x_test[:, 1:])
```



Vidyavardhini's College of Engineering and Technology

Department of Computer Engineering

Academic Year : 2023-24 (Odd Sem)



```
[30] print(x_test)

[[0.0 1.0 0.0 -1.4603817544038124 -0.900517004068727]
 [1.0 0.0 0.0 -0.4897366439180435 0.2056403793225396]]

[31] from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(x_train, y_train)

DecisionTreeClassifier

DecisionTreeClassifier(criterion='entropy', random_state=0)

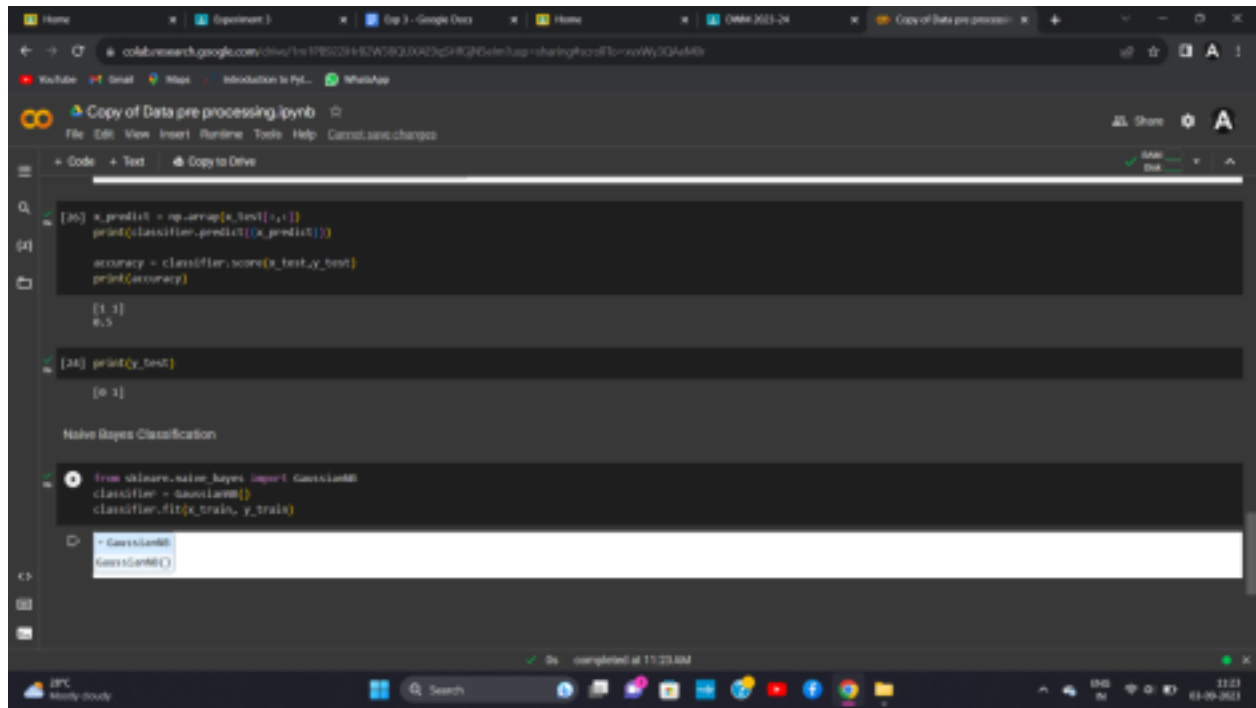
[30] x_predict = np.array(x_test[:, 1:])
print(classifier.predict(x_predict))

accuracy = classifier.score(x_test, y_test)
print(accuracy)

[1 1]
0.5

[30] print(y_test)

[0 1]
```



```
[36] x_predict = np.array(x_test[0:1])
      print(classifier.predict([x_predict]))

accuracy = classifier.score(x_test,y_test)
print(accuracy)

[1. 1]
0.5

[38] print(y_test)

[0 1]

Naive Bayes Classification

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

GaussianNB
GaussianNB()
```

**Conclusion:**Pre-processing is a critical step in the data analysis and machine learning pipeline, and its importance cannot be overstated. It involves a series of operations and



Vidyavardhini's College of Engineering and Technology

Department of Computer Engineering

Academic Year : 2023-24 (Odd Sem)

transformations applied to raw data before it is used for modeling or analysis. Here are some key reasons why preprocessing is essential and what can happen if data is not pre-processed:

**Data Quality Improvement:** Raw data often contains errors, missing values, outliers, and inconsistencies. Pre-processing helps identify and handle these issues, resulting in cleaner and more reliable data. Without pre-processing, erroneous or missing data can lead to incorrect conclusions and biased models.

**Feature Engineering:** Feature engineering is the process of creating new features or transforming existing ones to extract relevant information. Pre-processing plays a crucial role in feature engineering by enabling the creation of meaningful features from the raw data. Without it, models may not capture important patterns and relationships in the data.

**Normalization and Scaling:** Different features in a dataset may have different scales, which can affect the performance of many machine learning algorithms. Pre

processing techniques like normalization and scaling bring all features to a common scale, making it easier for models to learn and generalize patterns.

If data is not pre-processed properly:

**Model Performance Degradation:** Without appropriate pre-processing, models may struggle to learn from noisy or unstructured data, leading to lower predictive accuracy and generalization.

**Misleading Insights:** In data analysis, improper pre-processing can lead to misleading conclusions, as the underlying patterns in the data may not be accurately represented.

**Increased Computational Costs:** Raw, unprocessed data may require more complex and resource-intensive models to achieve reasonable performance, increasing computational costs.