Aim: Demonstrate various data Pre-Processing techniques for a given datasets.

Data Pre-processing Stage This notebook contains the basic data pre processing steps.

Preprocessing refers to the transformations applied to the data before feeding it to the machine learning algorithms. The data gathered from different sources is collected in raw format which is not feasible for the analysis. Data Preprocessing technique is used to convert the raw data into a clean data set.

Why preprocessing ?

Real world data are generally Incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data. Noisy: containing errors or outliers. Inconsistent: containing discrepancies in codes or names.

1. Data Cleaning/Cleansing

Real-world data tend to be incomplete, noisy, and inconsistent. Data Cleaning/Cleansing routines attempt to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data.

Data can be noisy, having incorrect attribute values. Owing to the following, the data collection instruments used may be fault. Maybe human or computer errors occurred at data entry. Errors in data transmission can also occur.

“Dirty” data can cause confusion for the mining procedure. Although most mining routines have some procedures, they deal incomplete or noisy data, which are not always robust. Therefore, a useful Data Preprocessing step is to run the data through some Data Cleaning/Cleansing routines.

2. Data Integration

Combining data from multiple sources.

Data Integration is involved in data analysis task which combines data from multiple sources into a coherent data store, as in data warehousing. These sources may include multiple databases, data cubes, or flat files. The issue to be considered in Data Integration is schema integration. It is tricky.

How can real-world entities from multiple data sources be ‘matched up’? This is referred as entity identification problem. For example, how can a data analyst be sure that customer_id in one database and cust_number in another refer to the same entity? The answer is metadata. Databases and data warehouses typically have metadata. Simply, metadata is data about data.

Metadata is used to help avoiding errors in schema integration. Another important issue is redundancy. An attribute may be redundant, if it is derived from another table. Inconsistencies in attribute or dimension naming can also cause redundancies in the resulting data set.

3. Data Transformation

Data are transformed into appropriate forms of mining. Data Transformation involves the following:

1. In Normalisation, where the attribute data are scaled to fall within a small specified range, such as -1.0 to 1.0, or 0 to 1.0.

2. Smoothing works to remove the noise from the data. Such techniques include binning, clustering, and regression.

3. In Aggregation, summary or aggregation operations are applied to the data. For example, daily sales data may be aggregated so as to compute monthly and annual total amounts. This step is typically used in constructing a data cube for analysis of the data at multiple granularities.

4. In Generalisation of the Data, low level or primitive/raw data are replaced by higher level concepts through the use of concept hierarchies. For example, categorical attributes are generalised to higher level concepts street into city or country. Similarly, the values for numeric attributes may be mapped to higher level concepts like, age into young, middle-aged, or senior.

4. Data Reduction

Reducing representation of data set.

Complex data analysis and mining on huge amounts of data may take a very long time, making such analysis impractical or infeasible. Data Reduction techniques are helpful in analysing the reduced representation of the data set without compromising the integrity of the original data and yet producing the qualitative knowledge. Strategies for data reduction include the following:

1. In Data Cube Aggregation, aggregation operations are applied to the data in the construction of a data cube.
2. In Dimension Reduction, irrelevant, weakly relevant, or redundant attributes or dimensions may be detected and removed.
3. In Data Compression, encoding mechanisms are used to reduce data set size. The methods used for Data Compression are Wavelet Transform and Principle Component Analysis.
4. In Numerosity Reduction, data is replaced or estimated by alternative and smaller data representations such as parametric models (which store only the model parameters instead of the actual data, e.g. Regression and Log-Linear Models) or non-parametric methods (e.g. Clustering, Sampling, and the use of histograms).
5. In Discretisation and Concept Hierarchy Generation, raw data values for attributes are replaced by ranges or higher conceptual levels. Concept hierarchies allow the mining of data at multiple levels of abstraction and are powerful tools for data mining.

Important Data Preprocessing Techniques

Now that you know more about the data preprocessing phase and why it's important, let's look at the main techniques to apply in the data, making it more usable for our future work. The techniques that we'll explore are:

Data Cleaning
Dimensionality Reduction
Feature Engineering
Sampling Data
Data Transformation
Imbalanced Data

Data Cleaning

One of the most important aspects of the data preprocessing phase is detecting and fixing bad and inaccurate observations from your dataset in order to improve its quality. This technique refers to identifying incomplete, inaccurate, duplicated, irrelevant or null values in the data. After identifying these issues, you will need to either modify or delete them. The strategy that you adopt depends on the problem domain and the goal of your project. Let's see some of the common issues we face when analyzing the data and how to handle them.

Noisy Data

Usually, noisy data refers to meaningless data in your dataset, incorrect records, or duplicated observations. For example, imagine there is a column in your database for 'age' that has negative values. In this case, the observation doesn't make sense, so you could delete it or set the value as null (we'll cover how to treat this value in the "Missing Data" section).

Another case is when you need to remove unwanted or irrelevant data. For example, say you need to predict whether a woman is pregnant or not. You don't need the information about their hair color, marital status or height, as they are irrelevant for the model.

An outlier can be considered noise, even though it might be a valid record, depending on the outlier. You'll need to determine if the outlier can be considered noise data and if you can delete it from your dataset or not.

Missing Data

Another common issue that we face in real-world data is the absence of data points. Most machine learning models can't handle missing values in the data, so you need to intervene and adjust the data to be properly used inside the model.

Structural Errors

Structural errors usually refer to some typos and inconsistencies in the values of the data.

For example, say that there is a marketplace and we sell shoes on our website. The data about the same product can be written in different ways by different sellers that sell the same shoes. Imagine that one of the attributes we have is the brand of the shoes, and aggregating the name of the brand for the same shoes we have: Nike, nike, NIKE. We need to fix this issue before giving this data to the model, otherwise, the model may treat them as different things. In this case, it's an easy fix: just transform all the words to lowercase. It may require more complex changes to fix inconsistencies and typos in other scenarios, though.

This issue generally requires manual intervention rather than applying some automated techniques.

Dimensionality Reduction

The dimensionality reduction is concerned with reducing the number of input features in training data.

The Curse of Dimensionality in Your Dataset

With a real-world dataset, there are usually tons of attributes, and if we don't reduce this number, it may affect the model's performance later when we feed it this dataset. Reducing the number of features while keeping as much variation in the dataset as possible will have a positive impact in many ways, such as:

Requiring less computational resources
Increasing the overall performance of the model
Preventing overfitting (when the model becomes too complex and the model memorizes the training data, instead of learning, so in the test data the performance decreases a lot)
Avoiding multicollinearity (high correlation of one or more independent variables). Also, applying this technique will reduce the noise data.

Let's dive into the main types of dimensionality reduction we can apply to our data to make it better for later use.

Feature Selection

Feature selection refers to the process of selecting the most important variables (features) related to your prediction variable, in other words, selecting the attributes which contribute most to your model. Ere are some techniques for this approach that you can apply either automatically or manually:

Correlation Between Features: This is the most common approach, which drops some features that have a high correlation with others.
Statistical Tests: Another alternative is to use statistical tests to select the features, checking the relationship of each feature individually with the output variable. There are many examples in the scikit-learn library like SelectKBest, SelectPercentile, chi2, f_classif, f_regression.
Recursive Feature Elimination (RFE**)**: The Recursive Feature Elimination, also known as Backward Elimination, where the algorithm trains the model with all features in the dataset, calculating the performance of the model, and then drops one feature at a time, stopping when the performance improvement becomes negligible.

Variance Threshold: Another feature selection method is the variance threshold, which detects features with high variability within the column, selecting those that got over the threshold. The premise of this approach is that features with low variability within themselves have little influence on the output variable.

Program:

```
import numpy as np
import pandas as pd
import os
#NumPy is module for Python. The name is an acronym for "Numeric Python" or "Numerical Python".
#The OS module in Python provides a way of using operating system dependent functionality.
#Pandas is an open-source Python Library providing high-performance data manipulation


-----------------------------------------------------------------------------------------------

dataset = pd.read_excel("/content/drive/MyDrive/Colab Notebooks/dataset /Country.xlsx")
print('Load the datasets...')


# Print the shape of the dataset
print ('dataset: %s'%(str(dataset.shape)))
```

O/p

```
Load the datasets... dataset: (15, 4)
```

Dataset

	Country	Age	Salary	Purchased
0	India	34.0	92000.0	Yes
1	Sri lanka	22.0	25000.0	Yes
2	China	31.0	74000.0	Yes
3	Sri lanka	29.0	NaN	No
4	China	55.0	98000.0	Yes
5	India	24.0	30000.0	No
6	Sri lanka	28.0	40000.0	No
7	India	NaN	60000.0	No
8	China	51.0	89000.0	Yes
9	India	44.0	78000.0	Yes
10	Sri lanka	21.0	20000.0	No
11	China	25.0	30000.0	Yes
12	India	33.0	45000.0	Yes
13	India	42.0	65000.0	Yes
14	Sri lanka	33.0	22000.0	No

```
-------------------------------------------------------------

df=pd.DataFrame(dataset)
df.duplicated()
```

O/p

0 False

1 False

2 False

3 False

4 False

5 False

6 False

7 False

8 False

9 False

10 False

11 False

12 False

13 False

14 False

```
# Separate the dependent and independent variables

# Independent variable
# iloc[rows,columns]
# Take all rows
# Take last but one column from the dataset (:-1)
X = dataset.iloc[:,:-1].values

# Dependent variable
# iloc[rows,columns]
# Take all rows
# Take last column from the dataset (:-1)
Y = dataset.iloc[:,3].values


# Print the X and Y
print ('X: %s'%(str(X)))
print ('-----------------------------------')
print ('Y: %s'%(str(Y)))
```

O/P

X: [['India' 34.0 92000.0]

['Sri lanka' 22.0 25000.0]

['China' 31.0 74000.0]

['Sri lanka' 29.0 nan]

['China' 55.0 98000.0]

['India' 24.0 30000.0]

['Sri lanka' 28.0 40000.0]

['India' nan 60000.0]

['China' 51.0 89000.0]

['India' 44.0 78000.0]

['Sri lanka' 21.0 20000.0]

['China' 25.0 30000.0]

['India' 33.0 45000.0]

['India' 42.0 65000.0]

['Sri lanka' 33.0 22000.0]]

Y: ['Yes' 'Yes' 'Yes' 'No' 'Yes' 'No' 'No' 'No' 'Yes' 'Yes' 'No' 'Yes' 'Yes' 'Yes' 'No']

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.

The sklearn.preprocessing package provides several common utility functions and transformer classes

to change raw feature vectors into a representation that is more suitable for the downstream estimators.

Imputer Class takes the follwing parameters:

missing_values : The missing values in our dataset are called as NaN (Not a number).Default is NaN

strategy : replace the missing values by mean/median/mode. Default is mean.

axis : if axis = 0, we take we of the column and if axis = 1, we take mean value of row.

```
from sklearn.impute import SimpleImputer
imputer=SimpleImputer(missing_values=np.nan,strategy='mean')
#from sklearn import impute
# Fit imputer for columns 1 and 2 of X matrix.
imputer=imputer.fit(X[:,2:4])
##Replace missing data with mean of column
X[:,2:4]=imputer.transform(X[:,2:4])
X
```

```
from sklearn.preprocessing import LabelEncoder
#Label Encoder:
#* It is used to transform non-numerical labels to numerical labels (or nominal categorical variables).
#* Numerical labels are always between 0 and n_classes-1.
lable_Encoder_X=LabelEncoder()
X[:,0]=lable_Encoder_X.fit_transform(X[:,0])
X
```

O/P

```
array([[1, 34.0, 92000.0], [2, 22.0, 25000.0], [0, 31.0, 74000.0], [2, 29.0, 54857.142857142855], [0, 55.0, 98000.0], [1,
24.0, 30000.0], [2, 28.0, 40000.0], [1, nan, 60000.0], [0, 51.0, 89000.0], [1, 44.0, 78000.0], [2, 21.0, 20000.0], [0, 25.0,
30000.0], [1, 33.0, 45000.0], [1, 42.0, 65000.0], [2, 33.0, 22000.0]], dtype=object)
```

Conclusion: