Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

Course: Minors in Data Science

Subject: Machine Learning (DJ19MN4C2)

AY: 2022-23

Experiment 2

(Data Pre-processing)

Name: Ayush Jain
SAP ID: 60004200132
Branch: Computer Engineering

Aim: Demonstrate various **data pre-processing** and visualisation techniques for a given dataset.

Theory:

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

Why do we need Data Preprocessing?

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data preprocessing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

Data preprocessing techniques

There are two main categories of preprocessing -- data cleansing and feature engineering. Each includes a variety of techniques, as detailed below.

Data cleansing

Techniques for cleaning up messy data include the following:

Identify and sort out missing data. There are a variety of reasons a data set might be missing individual fields of data. Data scientists need to decide whether it is better to discard records with missing fields, ignore them or fill them in with a probable value. For example, in an IoT application that records temperature, adding in a missing average temperature between the previous and subsequent record might be a safe fix.

Reduce noisy data. Real-world data is often noisy, which can distort an analytic or AI model. For example, a temperature sensor that consistently reported a temperature of 75 degrees Fahrenheit might erroneously report a temperature as 250 degrees. A variety of statistical approaches can be used to reduce the noise, including binning, regression and clustering.

Identify and remove duplicates. When two records seem to repeat, an algorithm needs to determine if the same measurement was recorded twice, or the records represent different events. In some cases, there may be slight differences in a record because one field was recorded incorrectly. In other cases, records that seem to be duplicates might indeed be different, as in a father and son with the same name who are living in the same house but should be represented as separate individuals. Techniques for identifying and removing or joining duplicates can help to automatically address these types of problems.

Feature engineering

Feature engineering, as noted, involves techniques used by data scientists to organize the data in ways that make it more efficient to train data models and run inferences against them. These techniques include the following:

Feature scaling or normalization. Often, multiple variables change over different scales, or one will change linearly while another will change exponentially. For example, salary might be measured in thousands of dollars, while age is represented in double digits. Scaling helps to transform the data in a way that makes it easier for algorithms to tease apart a meaningful relationship between variables.

Data reduction. Data scientists often need to combine a variety of data sources to create a new AI or analytics model. Some of the variables may not be correlated with a given outcome and can be safely discarded. Other variables might be relevant, but only in terms of relationship -- such as the ratio of debt to credit in the case of a model predicting the likelihood of a loan repayment; they may be combined into a single variable. Techniques like principal component analysis play a key role in reducing the number of dimensions in the training data set into a more efficient representation.

Discretization. It's often useful to lump raw numbers into discrete intervals. For example, income might be broken into five ranges that are representative of people who typically apply for a given type of loan. This can reduce the overhead of training a model or running inferences against it.

Feature encoding. Another aspect of feature engineering involves organizing unstructured data into a structured format. Unstructured data formats can include text, audio and video. For example, the process of developing natural language processing algorithms typically starts by using data transformation algorithms like Word2vec to translate words into numerical vectors. This makes it easy to represent to the algorithm that words like "mail" and "parcel" are similar,

while a word like "house" is completely different. Similarly, a facial recognition algorithm might reencode raw pixel data into vectors representing the distances between parts of the face.

Lab Assignments to complete in this session

Use the given dataset and perform the following tasks:

- **Getting the dataset**
- **Importing libraries**
- **Importing datasets**
- **Finding Missing Data**
- **Encoding Categorical Data**
- **Splitting dataset into training and test set**
- **Feature scaling**
- **Outlier detection using Box plots**

Dataset 1: country.csv

Getting the dataset and Importing libraries

```
# Getting the dataset and Importing libraries
import numpy as nm
import matplotlib.pyplot as mpt
import pandas as pd
```

Importing datasets

```
[58] dataset= pd.read_csv('/content/Data.csv')
```

Extracting Independent and Dependant value

```
X= dataset.iloc[:,:-1].values
Y= dataset.iloc[:,3].values
```

Handling missing data (Replacing missing data with the mean value)

```
# Handling missing data (Replacing missing data with the mean value)
dataset['Age'].fillna(dataset['Age'].mean(), inplace = True)
dataset['Salary'].fillna(dataset['Salary'].mean(), inplace = True)
```

Encoding Catgorical data for Country Variable

```
# Encoding categorical data
from sklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()
dataset['Country'] = labelencoder_X.fit_transform(dataset['Country'])
dataset['Purchased'] = labelencoder_X.fit_transform(dataset['Purchased'])
```

Splitting the Dataset into the Training set and Test set

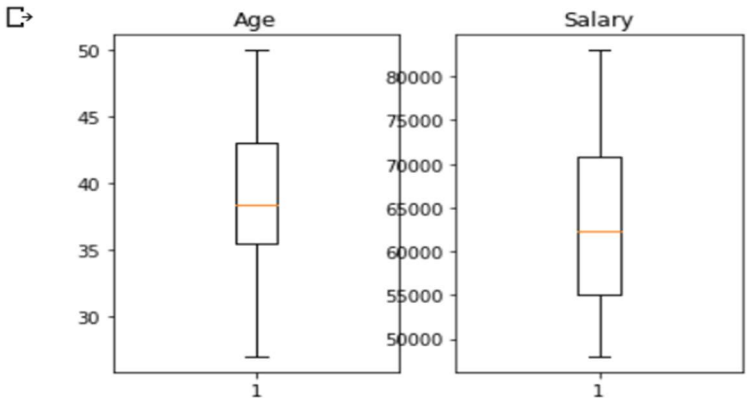
```
# Splitting dataset into training and test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Feature Scaling

```
# Feature scaling
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

Outlier detection using Box plots

```
# Outlier detection using box plots
fig, axs = mpt.subplots(1, 2)
axs[0].boxplot(dataset['Age'])
axs[0].set_title('Age')
axs[1].boxplot(dataset['Salary'])
axs[1].set_title('Salary')
mpt.show()
```



Conclusion: The above implementation task demonstrated some of the commonly used data pre-processing techniques in machine learning. By properly pre-processing the data, we can help the machine learning model to better understand the patterns in the data and make accurate predictions on unseen data.