Experiment No. 3

Title: Data Preprocessing Reading the Dataset Handling Missing Data Conversion to theTensorFormat

Aim: Data Preprocessing Reading the Dataset Handling Missing Data Conversion to theTensorFormat

Theory:

Data preprocessing:

Data preprocessing is a crucial step in the data analysis and machine learning pipeline. It involves cleaning, transforming, and preparing the data to be used in machine learning models. Let's go through the steps of data preprocessing, including reading the dataset, handling missing data, and converting the data into the tensor format.

Step 1) Reading the Dataset:

To read a dataset into Python, you can use various libraries such as NumPy, Pandas, or TensorFlow (tf.data) depending on the format of your data. Pandas is a popular choice for handling tabular data.

In [2]:
```
import pandas as pd

df = pd.read_csv('DataSet.csv') #Read a CSV file into a Pandas DataFrame
print(df)
```

```
      Name   Age  Gender  Salary
0    Alice  25.0  Female   50000
1      Bob  30.0    Male   60000
2  Charlie   NaN    Male   45000
3    David  28.0    Male   70000
4      Eva  35.0  Female   55000
```

Step 2) Handling Missing Data:

Missing data is a common issue in datasets. You need to handle missing data appropriately to avoid biased or incorrect results in your analysis or model training

In [3]:
```
print(df.isnull().sum()) #Check for missing values in the DataFrame

#Option 1: Drop rows with missing values
#df = df.dropna()

#Option 2: Fill missing values with a specific value (e.g. mean, median, or
mean_age = df['Age'].mean()
df['Age'].fillna(mean_age, inplace=True)
#df = pd.DataFrame(df)

df['Gender'] = df['Gender'].map({'Male':1, 'Female':0})
print(df)
```

```
Name      0
Age       1
Gender    0
Salary    0
dtype: int64
      Name   Age  Gender  Salary
0    Alice  25.0       0   50000
1      Bob  30.0       1   60000
2  Charlie  29.5       1   45000
3    David  28.0       1   70000
4      Eva  35.0       0   55000
```

Step 3) Conversion to the Tensor Format:

Machine learning models, especially those built with TensorFlow, often require data in tensor format. A tensor is a multi-dimensional array

In [6]:
```
import numpy as np
import tensorflow as tf


# Extract features and Labels from the DataFrame
X = df[{'Age', 'Gender'}].values
y = df['Salary'].values


#Convert to Tensorflow tensors
X_tensor = tf.constant(X, dtype=tf.float32)
y_tensor = tf.constant(y, dtype=tf.int32)


#Alternatively, you can use Nupy arrays directly as Tensorflow tensors
X_tensor = tf.convert_to_tensor(X, dtype=tf.float32)
y_tensor = tf.convert_to_tensor(y, dtype=tf.int32)


print("\n X tensor \n",X_tensor)
print("\n Y tensor \n",y_tensor)
```

```
 X tensor
 tf.Tensor(
[[ 0.   25. ]
 [ 1.   30. ]
 [ 1.   29.5]
 [ 1.   28. ]
 [ 0.   35. ]], shape=(5, 2), dtype=float32)

 Y tensor
 tf.Tensor([50000 60000 45000 70000 55000], shape=(5,), dtype=int32)
```

```
C:\Users\Sanika\AppData\Local\Temp\ipykernel_13576\2201011612.py:6: Future
Warning: Passing a set as an indexer is deprecated and will raise in a fut
ure version. Use a list instead.
  X = df[{'Age', 'Gender'}].values
```

Conclusion:

Successful installation of essential library, Data Preprocessing Reading the Dataset Handling Missing Data Conversion to the Tensor Format.