Lab 1 : Data Pre-Processing: Building Good Training Sets

**Note:-Link for Dataset:-
https://www.kaggle.com/competitions/titanic/data?select=train.csv

Procedure:-

The famous "Titanic: Machine Learning from Disaster" dataset from Kaggle as an example. Here's a Python program that demonstrates data pre-processing steps for building good training sets using the Titanic dataset:

Steps performed in the Program:-

1. The pandas library is used to load the dataset from the CSV file into a DataFrame.
2. Any necessary data cleaning steps can be performed. In this example, we remove unnecessary columns (e.g., 'PassengerId', 'Name', 'Ticket', 'Cabin') using the drop() function. Additionally, missing values in 'Age' and 'Embarked' columns are handled by filling them with the mean and mode values, respectively.
3. The dataset is split into features (X) and the target variable (y), where the 'Survived' column represents the target variable.
4. Label encoding is performed for categorical variables. Here, the LabelEncoder from scikit-learn is used to transform categorical values ('Sex' and 'Embarked') into numeric representations.
5. Feature scaling is applied to normalize the feature values. Here, the StandardScaler from scikit-learn is used to standardize the features.
6. The dataset is split into training and testing sets using the train_test_split() function from scikit-learn. In this example, 80% of the data is allocated to the training set and 20% to the testing set.
7. Finally, the shapes of the training and testing sets are printed to verify the sizes of the sets.

Program :-

```
import pandas as pd

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.model_selection import train_test_split
```

```
# Load the dataset
data = pd.read_csv('train.csv')

# Perform any necessary data cleaning steps
# Example: Removing unnecessary columns
data = data.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1)

# Handle missing values
data['Age'].fillna(data['Age'].mean(), inplace=True)
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)

# Split the dataset into features and labels
X = data.drop('Survived', axis=1)
y = data['Survived']

# Perform label encoding for categorical variables
encoder = LabelEncoder()
categorical_cols = ['Sex', 'Embarked']
for col in categorical_cols:
    X[col] = encoder.fit_transform(X[col])

# Perform feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)
```

```
# Print the shapes of the training and testing sets
print("Training set shape:", X_train.shape)
print("Testing set shape:", X_test.shape)
```

Output :-

```
Training set shape: (712, 7)
Testing set shape: (179, 7)
```

Result:

The Data Pre-Processing for building good training sets is executed successfully.