Project: Assessment of marginal workers in tamilnadu -A socioeconomic analysis

Detailed explanation of dataset:

A dataset related to marginal workers would specifically contain information about individuals who fall into the category of marginal workers. **Here is a more detailed explanation of the dataset:**

* \*Table Code:\* This column may indicate the unique identifier or code for the specific table or dataset being referred to.
* \*State Code:\* This column likely represents a unique identifier or code for different states or regions within a country.
* \*District Code:\* It's probably an identifier or code for various districts or subdivisions within the states.
* \*Area Name:\* This column likely contains the names or labels for specific areas, which could be districts, towns, or regions.
* \*Total/ Rural/ Urban:\* This is likely a categorical column that classifies the data into three categories: "Total," "Rural," and "Urban." It may represent different population segments or areas.
* \*Age group:\* This column may contain information about different age groups or ranges, used for demographic analysis.
* \*Worked for 3 months or more but less than 6 months - Persons/Males/Females:\* These columns likely contain data about the number of persons, males, and females who have worked for a specific duration (between 3 and 6 months).
* \*Worked for less than 3 months - Persons/Males/Females:\* Similar to the previous columns, these represent data about the number of persons, males, and females who have worked for less than 3 months.

Implementation of dataset:

Implementing marginal workers involves several steps using programming language like python and popular libraries such as scikit-learn below a simplified example of how you might implement the marginal workers in tamilnadu using python

import pandas as pd

import numpy as np

# Define the number of samples

num\_samples = 100

# Create synthetic data

data = {

'Table Code': np.random.randint(1000, 2000, size=num\_samples), # Random table codes

'State Code': np.random.randint(1, 30, size=num\_samples), # Random state codes

'District Code': np.random.randint(1, 100, size=num\_samples), # Random district codes

'Area Name': np.random.choice(['Rural', 'Urban'], num\_samples), # Random area names

'How many Months Worked': np.random.randint(0, 12, size=num\_samples), # Random months worked in a year

'ID': range(1, num\_samples + 1),

'Age': np.random.randint(18, 60, size=num\_samples),

'Gender': np.random.choice(['Male', 'Female'], num\_samples),

'Education Level': np.random.choice(['Primary', 'Secondary', 'Higher Secondary', 'Graduate'], num\_samples),

'Employment Type': np.random.choice(['Agriculture', 'Construction', 'Informal Labor'], num\_samples),

'Monthly Income': np.random.randint(3000, 25000, size=num\_samples),

}

# Create a DataFrame

df = pd.DataFrame(data)

# Save the dataset to a CSV file

df.to\_csv('marginal\_workers\_dataset.csv', index=False) # Save the dataset to a CSV file without including the index

# Provide a summary of the dataset

print("Synthetic Marginal Workers Dataset:")

print(df.head()) # Display the first few rows of the generated dataset

**Program to load the dataset:**

import pandas as pd

# Step 1: Define the path to your CSV file

csv\_file\_path = 'marginal\_workers\_dataset.csv' # Replace with the actual file path

# Step 2: Load the data from the CSV file

data = pd.read\_csv(csv\_file\_path)

# Step 3: Display basic information about the dataset

print("Basic Information about the Dataset:")

print(data.info())

# Step 4: Display the first few rows of the dataset

print("\nFirst Few Rows of the Dataset:")

print(data.head())

Preprocessing of data:

Data preprocessing is a critical step in preparing your data for analysis, modeling, or machine learning. It involves cleaning, transforming, and organizing the data to ensure that it is suitable for your specific objectives. Here's an explanation of the key steps in data preprocessing:

1. \*Data Collection:\* Gather data from various sources, which may include surveys, databases, text files, or APIs.

2. \*Data Cleaning:\*

- Handle Missing Values: Identify and deal with missing data. You can either remove rows with missing values, fill them in with appropriate values, or use imputation techniques.

- Remove Duplicates: Check for and remove duplicate records or rows to ensure data integrity.

3. \*Data Transformation:\*

- Data Encoding: Convert categorical data into a numerical format. One-hot encoding and label encoding are common methods.

- Scaling/Normalization: Ensure that numeric features are on the same scale. Techniques like min-max scaling and standardization (z-score scaling) are used.

- Feature Engineering: Create new features or transform existing ones to improve the quality of data for modeling.

- Binning: Group continuous data into bins or categories.

- Log Transformation: Apply logarithmic transformations to normalize data distributions.

4. \*Data Reduction:\*

- Dimensionality Reduction: Use techniques like Principal Component Analysis (PCA) or feature selection to reduce the number of features while retaining essential information.

- Outlier Detection: Identify and handle outliers to prevent them from skewing the analysis.

5. \*Data Splitting:\*

- Split the dataset into training and testing subsets for machine learning tasks. This ensures that the model's performance is evaluated on unseen data.

6. \*Data Scaling:\* Scale or normalize numerical features to ensure they have a similar range. This is important for many machine learning algorithms.

7. \*Data Standardization:\* Standardize data so that it has a mean of 0 and a standard deviation of 1. This is particularly useful for algorithms that rely on distances or gradients, such as gradient descent.

8. \*Handling Imbalanced Data:\* If your dataset has imbalanced classes (e.g., more non-marginal workers than marginal workers), you may need to address this issue to avoid bias in modeling results.

9. \*Data Sampling:\* In cases of large datasets, you might perform random sampling to create smaller representative subsets.

10. \*Feature Scaling:\* Standardize or normalize features to ensure that no feature has undue influence on a model.

Data preprocessing can be a time-consuming and iterative process, but it is crucial for ensuring that the data is of high quality and that any analysis or models built on it are reliable and accurate. The specific steps you perform may vary depending on your dataset and the goals of your analysis or modeling.

**Program to preprocess the data:**

**import pandas as pd**

**from sklearn.preprocessing import LabelEncoder, StandardScaler**

**from sklearn.model\_selection import train\_test\_split**

**# Step 1: Load the dataset**

**data = pd.read\_csv('marginal\_workers\_dataset.csv') # Replace 'marginal\_workers\_dataset.csv' with your dataset file path**

**# Step 2: Handle Missing Values (if any)**

**# Example: Fill missing values in 'Monthly Income' with the mean**

**mean\_income = data['Monthly Income'].mean()**

**data['Monthly Income'].fillna(mean\_income, inplace=True)**

**# Step 3: Encode Categorical Variables**

**# Example: Encode 'Gender' and 'Area Name' using label encoding**

**label\_encoder = LabelEncoder()**

**data['Gender'] = label\_encoder.fit\_transform(data['Gender'])**

**data['Area Name'] = label\_encoder.fit\_transform(data['Area Name'])**

**# Step 4: Split the Data into Training and Testing Sets**

**X = data.drop(columns=['Monthly Income']) # Features (excluding 'Monthly Income')**

**y = data['Monthly Income'] # Target variable**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)**

**# Step 5: Feature Scaling**

**# Example: Standardize the features using StandardScaler**

**scaler = StandardScaler()**

**X\_train = scaler.fit\_transform(X\_train)**

**X\_test = scaler.transform(X\_test)**

**# Step 6: Perform any additional preprocessing steps (e.g., handling outliers, dimensionality reduction)**

**# Step 7: You can now use the preprocessed data for modeling or analysis**

**# Display a summary of the preprocessed data (optional)**

**print("Preprocessed Data Summary:")**

**print(X\_train[:5]) # Display the first 5 rows of the preprocessed features**

**print(y\_train[:5]) # Display the first 5 values of the target variable**

Performing Analysis In the Dataset:

This code demonstrates the following:

1.Loading the Dataset: Load the preprocessed dataset. You should replace 'marginal\_workers\_dataset.csv' with the actual file path.

2.Descriptive Statistics: Calculate and print the average age and average monthly income of the marginal workers in the dataset.

3.Data Visualization: Create visualizations to explore the data.

-A histogram shows the distribution of ages.

-A bar chart displays the distribution of gender in the dataset.

**Example Code:**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**# Step 1: Load the dataset (assuming you've already preprocessed it)**

**data = pd.read\_csv('marginal\_workers\_dataset.csv')**

**# Step 2: Descriptive Statistics**

**# Calculate basic statistics for 'Age' and 'Monthly Income'**

**age\_mean = data['Age'].mean()**

**income\_mean = data['Monthly Income'].mean()**

**print(f"Average Age: {age\_mean:.2f}")**

**print(f"Average Monthly Income: {income\_mean:.2f}")**

**# Step 3: Data Visualization**

**# Create a histogram for 'Age' and a bar chart for 'Gender'**

**plt.figure(figsize=(12, 5))**

**# Histogram for Age**

**plt.subplot(1, 2, 1)**

**plt.hist(data['Age'], bins=20, edgecolor='k')**

**plt.xlabel('Age')**

**plt.ylabel('Frequency')**

**plt.title('Age Distribution')**

**# Bar chart for Gender**

**plt.subplot(1, 2, 2)**

**gender\_counts = data['Gender'].value\_counts()**

**gender\_counts.plot(kind='bar', color='skyblue', edgecolor='k')**

**plt.xlabel('Gender')**

**plt.ylabel('Count')**

**plt.title('Gender Distribution')**

**plt.tight\_layout()**

**plt.show()**

**Sample Output:**

**Average Age: 37.28**

**Average Monthly Income: 11517.22**