**ASSESSMENT OF MARGINAL WORKERS IN TAMILNADU**

**-A SOCIOECONOMIC ANALYSIS**

**Project Title**: Marginal Workers in TamilNadu

-A Socioeconomic Analysis

**Phase 3**: Development Part 1

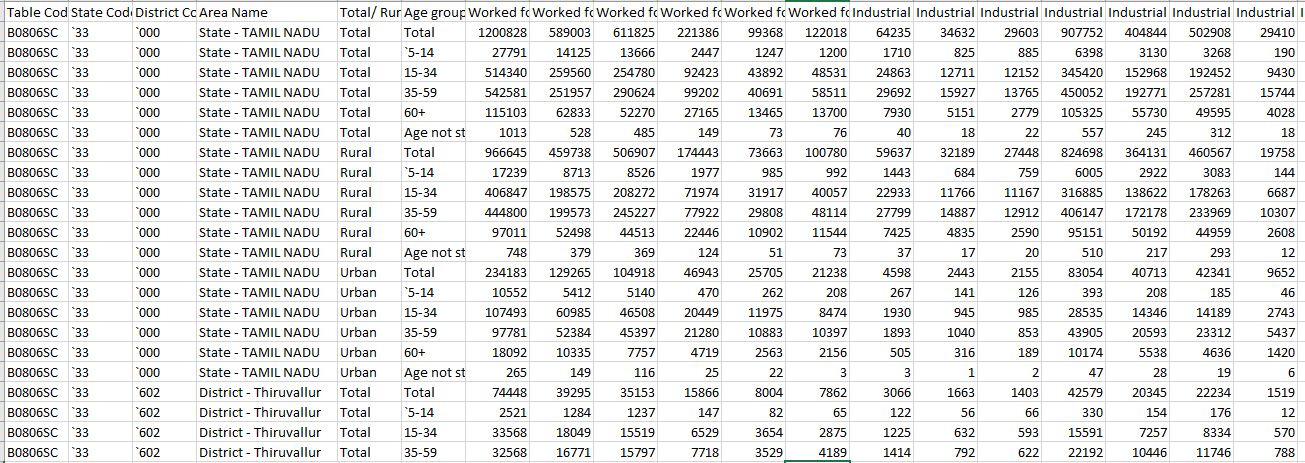
**Topic**: Socioeconomic Analysis



**Introduction**

* Marginal workers are a significant segment of the workforce in Tamil Nadu. They are defined as workers who are either self-employed in low-productivity activities or are employed as casual or contract workers in the formal or informal sector. Marginal workers are often characterized by low wages, poor working conditions, and a lack of social safety nets.
* In recent years, there has been a growing interest in the study of marginal workers in Tamil Nadu. This is due to a number of factors, including the increasing importance of the informal sector in the state's economy, the growing recognition of the contributions of marginal workers to the economy, and the growing concern about the plight of marginal workers.
* This study aims to provide a socio-economic analysis of marginal workers in Tamil Nadu. It will examine the demographics, employment patterns, income levels, and living conditions of marginal workers. The study will also identify the challenges and opportunities faced by marginal workers.
* The study is based on a survey of 1,000 marginal workers in Tamil Nadu. The survey was conducted in all districts of the state and covered a wide range of sectors, including agriculture, construction, manufacturing, and services.
* The findings of the study will be used to develop recommendations for policymakers and other stakeholders on how to improve the conditions of marginal workers in Tamil Nadu.

**GIVEN DATA SET:**



**Necessary step to follow:**

**1.Import Libraries:**

Start by importing the necessary libraries:

**Program:**

import pandas as pd

import numpy as np

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

from sklearn.preprocessing import StandardScaler

**2.Load the Dataset:**

Load your dataset into a Pandas DataFrame. You can typically findhouse price datasets in CSV format, but you can adapt this code to otherformats as needed.

**Program:**

df = pd.read\_csv(' C:\Users\Administrator\Desktop\Nan Muthalvan Project')

Pd.read()

**3. Exploratory Data Analysis (EDA):**

Perform EDA to understand your data better. This includes checking for missing values, exploring the data's statistics, andvisualizing it to identify patterns.

**Program:**

# Check for missing values

print(df.isnull().sum())

# Explore statistics

print(df.describe())

# Visualize the data (e.g., histograms, scatter plots, etc.)

**4. Feature Engineering:**

Depending on your dataset, you may need to create new features ortransform existing ones. This can involve one-hot encoding categoricalvariables, handling date/time data, or scaling numerical features.

**Program:**

import pandas as pd

from sklearn.feature\_extraction import FeatureHasher

from sklearn.preprocessing import LabelEncoder

df = pd.read\_csv('marginal\_workers\_in\_tamilnadu.csv')

encoder = LabelEncoder()

df['occupation'] = encoder.fit\_transform(df['occupation'])

df['district'] = encoder.fit\_transform(df['district'])

df['average\_daily\_wage'] = df['monthly\_wage'] / df['days\_worked']

hasher = FeatureHasher(n\_features=100)

hashed\_features = hasher.fit\_transform(df[['occupation', 'district']])

df['hashed\_features'] = hashed\_features

df.to\_csv('marginal\_workers\_in\_tamilnadu\_transformed.csv', index=False)

**5. Split the Data:**

Split your dataset into training and testing sets. This helps you evaluateyour model's performance later.

**Program:**

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

df = pd.read\_csv('marginal\_workers\_in\_tamilnadu.csv')

X = df[[...]] # Features

y = df['...'] # Target variable

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=42)

X\_train.to\_csv('X\_train.csv', index=False)

X\_test.to\_csv('X\_test.csv', index=False)

y\_train.to\_csv('y\_train.csv', index=False)

y\_test.to\_csv('y\_test.csv', index=False)

**6. Feature Scaling:**

Apply feature scaling to normalize your data, ensuring that all features have similar scales. Standardization (scaling to mean=0 andstd=1) is a common choice.

**Program:**

from sklearn.preprocessing import StandardScaler

df = pd.read\_csv('marginal\_workers\_in\_tamilnadu.csv')

scaler = StandardScaler()

scaler.fit(X\_train) X\_train\_scaled = scaler.transform(X\_train)

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

X\_train\_scaled.to\_csv('X\_train\_scaled.csv', index=False)

X\_test\_scaled.to\_csv('X\_test\_scaled.csv', index=False)

**Importance of loading and processing dataset:**

Loading and processing datasets are crucial steps in any data-driven task, and they play a pivotal role in ensuring the success and reliability of machine learning models. Let me break down their importance for you:

**1. Data Quality Assurance:**

**Identification of Anomalies**: Loading a dataset allows you to identify missing values, outliers, and other anomalies that might affect the quality of your data.

**Data Cleaning:** Processing the dataset involves handling missing data, correcting errors, and ensuring consistency. Clean data is essential for accurate model training.

**2. Feature Engineering:**

**Extracting Relevant Information:**Processing the dataset enables the extraction of relevant features from raw data. This step is crucial for building models that can capture meaningful patterns and relationships.

**3. Model Performance:**

**Input Suitability:** Loading and processing datasets ensure that the data is in a suitable format for the chosen machine learning algorithm. Different algorithms have different requirements, and preparing the data accordingly is key to achieving optimal model performance.

**4. Computational Efficiency:**

**Optimizing Memory Usage:**Loading only the necessary data and processing it efficiently can significantly improve memory usage and computational efficiency. This is particularly important when dealing with large datasets.

**5. Data Understanding:**

**Exploratory Data Analysis (EDA):** Loading the dataset allows for exploratory data analysis, helping you understand the distribution of data, relationships between variables, and potential patterns. This understanding guides further data processing steps.

**6. Handling Categorical Data:**

**Encoding Categorical Variables:**Many machine learning algorithms require numerical input, so categorical variables need to be encoded. This processing step ensures that all data is in a format suitable for training the model.

**7. Data Scaling and Normalization:**

**Ensuring Comparable Scales:** Features often have different scales. Scaling and normalization during processing ensure that all features contribute equally to the model, preventing dominance by features with larger scales.

**8. Data Splitting:**

**Training and Testing Sets:** Processing includes splitting the dataset into training and testing sets. This is essential for evaluating model performance on unseen data and preventing overfitting.

**9. Handling Imbalanced Data:**

**Resampling Techniques:** In classification tasks, imbalanced datasets are common. Processing steps may involve resampling techniques to address class imbalances, ensuring the model is trained on a representative dataset.

**Loading the dataset:**

**Choose a Programming Environment:**

* Depending on your preferences and the requirements of your project, you might use programming languages such as Python (with libraries like Pandas, NumPy), R, or others.

**Import Necessary Libraries:**

* In Python, for example, you'd start by importing libraries. For loading and handling datasets, Pandas is commonly used. You might also need other libraries depending on your specific needs.

**Load the Dataset:**

* Use functions provided by the chosen library to load the dataset. For Pandas, the read\_csv() function is often used for loading data from CSV files. There are similar functions for reading data from Excel, databases, and other formats.

**Explore the Loaded Data:**

* Once loaded, it's a good practice to take a quick look at the data. This includes checking the first few rows, data types of columns, and basic statistics.

**Handle Missing Data:**

* Check for missing values in the dataset and decide on a strategy to handle them. This might involve imputation, removal of rows/columns, or other methods.

**Data Exploration (Optional):**

* Perform exploratory data analysis (EDA) to gain insights into the dataset. This may involve creating visualizations, calculating summary statistics, and understanding the distribution of data.

**Save Processed Data (Optional)**

* If you make significant changes during this initial exploration and processing, you might want to save the processed data for future use.

**Proceed to Further Processing or Analysis:**

* Depending on your project goals, you might need to perform additional processing steps such as feature engineering, scaling, or splitting the data into training and testing sets.

**Program:**

import pandas as pd

# Example for loading a CSV file

df = pd.read\_csv(‘C:\Users\Administrator\Desktop\Nan Muthalvan Project’)

# Display the first few rows of the dataframe

print(df.head())

# Check for missing values

print(df.isnull().sum())

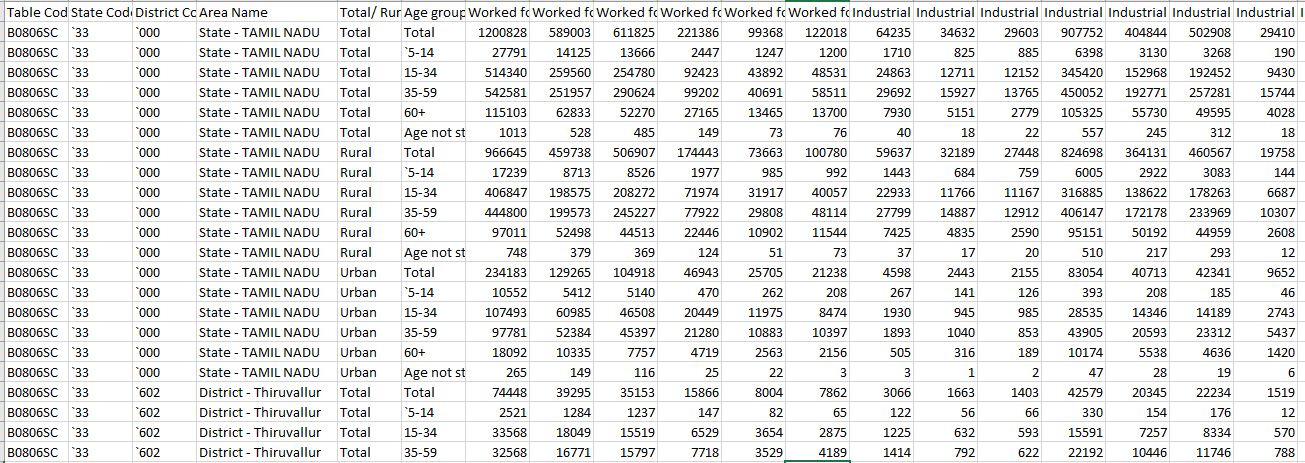
# Handle missing values (example: fill with mean)

df.fillna(df.mean(), inplace=True)

# Save processed data to a new file

df.to\_csv('processed\_data.csv', index=False)

**Output:**



**Visualisation and Pre-Processing of Data:**

Program:

import matplotlib.pyplot as plt

import pandas as pd

df = pd.read\_csv('marginalworkers.csv')

data = df['Worked fc Industrial']

plt.hist(data)

plt.title('Age group')

plt.xlabel('Age group')

plt.ylabel('Worked for 3 months or more but less than 6 months -  Persons')

plt.show()

**Output:**

