

ASSESSMENT OF MARGINAL WORKERS IN TAMILNADU-A SOCIOECONOMIC ANALYSIS

# Dataset and its detail explanation:

Dataset are taken from the skillup:

(i.e.) [https://tn.data.gov.in/catalog/marginal-workers-](https://tn.data.gov.in/catalog/marginal-workers-classified-age-industrial-category-and-sex-census-2011-india-and-states) [classified-age-industrial-category-and-sex-census-2011-india-](https://tn.data.gov.in/catalog/marginal-workers-classified-age-industrial-category-and-sex-census-2011-india-and-states) [and-states](https://tn.data.gov.in/catalog/marginal-workers-classified-age-industrial-category-and-sex-census-2011-india-and-states)

A "marginal workers dataset" typically refers to a collection of data that provides information about marginal workers in a particular context. Marginal workers are individuals who are employed, but their employment conditions are considered precarious or marginal, often with low wages, irregular work, and limited job security.

Such a dataset may include various attributes and information about these workers, such as:

* + 1. Age group
    2. Worked for 3 months or more but less than 6 months persons
    3. Worked for 3 months or more but less than 6 months males
    4. Worked for 3 monts or more but less than 6 months females
    5. Worked for less than 3 months persons
    6. Worked for less than 3 months males
    7. Worked for less than 3 months females Etc.,

This dataset can be used for various research and analysis, such as understanding the socio-economic conditions of marginal workers, identifying labor market trends, and developing policies to improve the well-being of these

workers. The specific details and scope of a marginal workers dataset can vary depending on the source and purpose of the data collection.

Implementations

How do you implement a dataset? Steps to Constructing Your Dataset

1. Collect the raw data.
2. Identify feature and label sources.
3. Select a sampling strategy.

Split the data.

# Begin building the project by LOAD the dataset.

To import a CSV file and put the contents into a Pandas dataframe we use the read\_csv() function.

import pandas as pd import numpy as np

a=pd.read\_csv("C:\\Users\\OOAD

LAB\\Downloads\\DDW\_B06ST\_3300\_State\_TAMIL\_NADU-2011.csv") print(a)

# PREPROCESS DATASET:

At the heart of Machine Learning is to process data.

Your **machine learning tools are as good as the quality of your data**. This blog deals with the various steps of **cleaning data**. Your data needs to go through a few steps before it is could be used for making predictions.

various steps of **cleaning data**. Your data needs to go through a few steps before it is could be used for making predictions.

**Steps involved in data preprocessing :**

* + 1. Importing the required Libraries
    2. Importing the data set
    3. Handling the Missing Data.
    4. Encoding Categorical Data.
    5. Splitting the data set into test set and training set.
    6. Feature Scaling.

So let us look at these steps one by one.

## Step 1: Importing the required Libraries:

to follow along you will need to download this dataset: [**Data.csv**](https://github.com/afrozchakure/Internity-Summer-Internship-Work/tree/master/Blogs/Preprocessing%22/t%20%22_blank)

Every time we make a new model, we will require to import Numpy and Pandas. Numpy is a Library which contains Mathematical functions and is used for scientific computing while Pandas is used to import and manage the data sets.

import pandas as pd import numpy as np

Here we are importing the pandas and Numpy library and

assigning a shortcut “pd” and “np” respectively.

## Step 2: Importing the Dataset

Data sets are available in .csv format. A CSV file stores tabular data in plain text. Each line of the file is a data record. We use the read\_csv method of the pandas library to read a local CSV file as a **dataframe**.

dataset = pd.read\_csv('Data.csv') After carefully inspecting our dataset, we are going to create a matrix of features in our dataset (X) and create a dependent vector (Y) with their respective observations. To read the columns, we will use iloc of pandas (used to fix the indexes for selection) which takes two parameters — [row selection, column selection].

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

## Step 3: Handling the Missing Data

An example of Missing data and Imputation

The data we get is rarely homogenous. Sometimes data can be missing and it needs to be handled so that it does not reduce the performance of our machine learning model.

To do this we need to replace the missing data by the Mean or Median of the entire column. For this we will be using the sklearn.preprocessing Library which contains a class called Imputer which will help us in taking care of our missing data.

from sklearn.preprocessing import Imputer

imputer = Imputer(missing\_values = "NaN", strategy = "mean", axis = 0)

Our object name is **imputer.** The Imputer class can take

parameters like :

1. **missing\_values** : It is the placeholder for the missing values. All occurrences of missing\_values will be imputed. We can give it an integer or “NaN” for it to find missing values.
2. **strategy** : It is the imputation strategy — If “mean”, then replace missing values using the mean along the axis (Column). Other strategies include “median” and “most\_frequent”.
3. **axis** : It can be assigned 0 or 1, 0 to impute along columns and 1 to impute along rows.

Now we fit the imputer object to our data.

imputer = imputer.fit(X[:, 1:3]) Now replacing the missing values with the mean of the column by using transform method.

X[:, 1:3] = imputer.transform(X[:, 1:3])

## Step 4: Encoding categorical data

Converting Categorical data into dummy variables. Any variable that is not quantitative is categorical. Examples include Hair color, gender, field of study, college attended, political affiliation, status of disease infection.

But why encoding ?

We cannot use values like “Male” and “Female” in mathematical equations of the model so we need to encode these variables into numbers.

To do this we import “LabelEncoder” class from

“sklearn.preprocessing” library and create an object labelencoder\_X of the LabelEncoder class. After that we use the fittransform method on the categorical features.

After Encoding it is necessary to distinguish between between the variables in the same column, for this we will use OneHotEncoder class from sklearn.preprocessing library.

## One-Hot Encoding

One hot encoding transforms categorical features to a format that works better with classification and regression algorithms. from sklearn.preprocessing import LabelEncoder,

OneHotEncoder labelencoder\_X = LabelEncoder() X[:, 0] = labelencoder\_X.fit\_transform(X[:, 0])

onehotencoder = OneHotEncoder(categorical\_features = [0]) X = onehotencoder.fit\_transform(X). toarray()labelencoder\_y = LabelEncoder()

y = labelencoder\_y.fit\_transform(y)

## Step 5: Splitting the Data set into Training set and Test Set

Now we divide our data into two sets, one for training our model called the **training set** and the other for testing

the performance of our model called the **test set**. The split is generally 80/20. To do this we import the “train\_test\_split” method of “sklearn.model\_selection” library.

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

Now to build our training and test sets, we will create 4 sets:

1. **X\_train** (training part of the matrix of features),
2. **X\_test** (test part of the matrix of features),
3. **Y\_train** (training part of the dependent variables associated with the X train sets, and therefore also the same indices) ,
4. **Y\_test** (test part of the dependent variables associated with the X test sets, and therefore also the same indices).

We will assign to them the test\_train\_split, which takes the parameters — arrays (X and Y), test\_size (Specifies the ratio in which to split the data set).

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split( X , Y , test\_size = 0.2, random\_state = 0)

## Step 6: Feature Scaling

Most of the machine learning algorithms use

the **Euclidean distance** between two data points in their computations . Because of this, **high magnitudes features will weigh more** in the distance calculations **than features with low magnitudes**. To avoid this Feature standardization or Z-score normalization is used. This is done by using

“StandardScaler” class of “sklearn.preprocessing”.

from sklearn.preprocessing import StandardScaler sc\_X = StandardScaler()

example:

#to import libraries

import pandas as pd import numpy as np

#to read the datasets

data=pd.read\_csv("C:\\Users\\OOAD

LAB\\Downloads\\DDW\_B06ST\_3300\_State\_TAMIL\_NADU-2011.csv") print(data)

#to handle the missing data:

data.isna()

from sklearn.impute import SimpleImputer

# to handle the dataset with missing values

data = np.array([[1, 2, np.nan], [4, np.nan, 6], [7, 8, 9]])

# Create an instance of the SimpleImputer class

imputer = SimpleImputer(strategy='mean')  # Other strategies: 'median', 'most\_frequent', 'constant'

# Fit the imputer to the data and transform it

imputed\_data = imputer.fit\_transform(data)

# The missing values have been replaced

print("Original Data:")

print(data)

print("\nImputed Data:")

print(imputed\_data)

#encoding the categorical dataset:

from sklearn.preprocessing import OneHotEncoder

import numpy as np

import pandas as pd

a=pd.read\_csv("C:\\Users\\OOAD LAB\\Downloads\\DDW\_B06ST\_3300\_State\_TAMIL\_NADU-2011.csv")

# Sample data

data = ['Age group','Worked for less than 3 months - Persons','Industrial Category - P to Q - Females']

# Initialize the OneHotEncoder

onehot\_encoder = OneHotEncoder(sparse=False)  # Use sparse=False to get a dense matrix

# Fit and transform the data (reshape is necessary)

encoded\_data = onehot\_encoder.fit\_transform(np.array(data).reshape(-1, 1))

print("Original data:", data)

print("One-Hot Encoded data:")

print(encoded\_data)

**#feature scalling process:**

from sklearn.preprocessing import StandardScaler

import numpy as np

a=pd.read\_csv("C:\\Users\\OOAD LAB\\Downloads\\DDW\_B06ST\_3300\_State\_TAMIL\_NADU-2011.csv")

# Sample data

data= np.array(mydataset["Worked for less than 3 months - Persons"]).reshape(-1,1)

# Initialize the StandardScaler

scaler = StandardScaler()

# Fit and transform the data

scaled\_data = scaler.fit\_transform(data)

# Print the scaled data

print("Original data:")

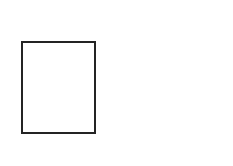
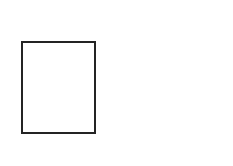
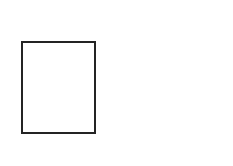
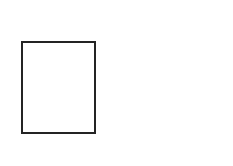
print(data)

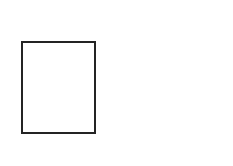
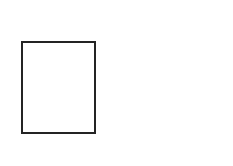
print("Scaled data:")

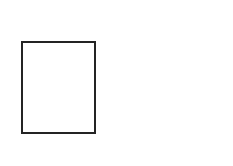
print(scaled\_data)

# Performing different analysis model:

Data analysis is an aspect of data science and data analytics that is all about analyzing data for different kinds of purposes. The data analysis process involves inspecting, cleaning, transforming and modeling data to draw useful insights from it.

Descriptive analysis. Diagnostic analysis. Exploratory analysis. Inferential analysis.

Predictive analysis. Causal analysis.

Mechanistic analysis.

## DESCRIPTIVE ANALYSIS

Descriptive analysis is the very first analysis performed in the data analysis process.

It generates simple summaries about samples and measurements.

It involves common, descriptive statistics like measures of central tendency, variability, frequency and position.

## DIAGNOSTIC ANALYSIS

Diagnostic analysis seeks to answer the question “Why did this happen?” by taking a more in-depth look at data to uncover subtle patterns. Here’s what you need to know:

Diagnostic analysis typically comes after descriptive analysis, taking initial findings and investigating why certain patterns in data happen.

Diagnostic analysis may involve analyzing other related data sources, including past data, to reveal more insights into current data trends.

Diagnostic analysis is ideal for further exploring patterns in data to explain anomalies.

## EXPLORATORY ANALYSIS (EDA):

Exploratory analysis involves examining or exploring data and finding relationships between variables that were previously unknown. Here’s what you need to know:

EDA helps you discover relationships between measures in your data, which are not evidence for the existence of the correlation, as denoted by the phrase, “Correlation doesn’t imply causation”.

It’s useful for discovering new connections and forming hypotheses. It drives design planning and data collection.

## INFERENTIAL ANALYSIS

Inferential analysis involves using a small sample of data to infer information about a larger population of data.

The goal of statistical modeling itself is all about using a small amount of information to extrapolate and generalize information to a larger group. Here’s what you need to know:

Inferential analysis involves using estimated data that is representative of a population and gives a measure of uncertainty or standard deviation to your estimation.

The accuracy of inference depends heavily on your sampling scheme. If the sample isn’t representative of the population, the generalization will be inaccurate. This is known as the central limit theorem.

## PREDICTIVE ANALYSIS

Predictive analysis involves using historical or current data to find patterns and make predictions about the future. Here’s what you need to know:

The accuracy of the predictions depends on the input variables.

Accuracy also depends on the types of models. A linear model might work well in some cases, and in other cases it might not.

Using a variable to predict another one doesn’t denote a causal relationship.

## CAUSAL ANALYSIS

Causal analysis looks at the cause and effect of relationships between variables and is focused on finding the cause of a correlation. Here’s what you need to know:

To find the cause, you have to question whether the observed correlations driving your conclusion are valid. Just looking at the surface data won’t help you discover the hidden mechanisms underlying the correlations.

Causal analysis is applied in randomized studies focused on identifying causation.

Causal analysis is the gold standard in data analysis and scientific studies where the cause of phenomenon is to be extracted and singled out, like separating wheat from chaff.

Good data is hard to find and requires expensive research and studies. These studies are analyzed in aggregate (multiple groups), and the observed relationships are just average effects (mean) of the whole population. This means the results might not apply to everyone.

## MECHANISTIC ANALYSIS

Mechanistic analysis is used to understand exact changes in variables that lead to other changes in other variables. Here’s what you need to know:

It’s applied in physical or engineering sciences, situations that require high precision and little room for error, only noise in data is measurement error.

It’s designed to understand a biological or behavioral process, the pathophysiology of a disease or the mechanism of action of an intervention.

## PRESCRIPTIVE ANALYSIS

Prescriptive analysis compiles insights from other previous data analyses and determines actions that teams or companies can take to prepare for predicted trends. Here’s what you need to know:

Prescriptive analysis may come right after predictive analysis, but it may involve combining many different data analyses.

Companies need advanced technology and plenty of resources to conduct prescriptive analysis. AI systems that process data and adjust automated tasks are an example of the technology required to perform prescriptive analysis.

# STEPS FOR DATA PROCESSING:

