ADS_PHASE3

ASSESSMENT OF MARGINAL WORKERS IN TAMILNADU-A SOCIOECONOMIC ANALYSIS

1. DATASET AND ITS EXPLANATION:

In our domain (APPLIED DATA SCIENCE), we are using the dataset which has been given by skill up team. Marginal Workers Classified by Age, Industrial Category and Sex, Census 2011 - India and States | Open Government Data (OGD) Platform India

A marginal worker is typically defined as a person who is engaged in irregular or occasional employment, or who works for a limited number of hours per day or days per week. This term is often used in the context of labour statistics and economic analysis to distinguish individuals who have less stable or consistent employment compared to regular or full-time workers. Marginal workers may include part-time employees, casual labourers, seasonal workers, or those who work intermittently. The concept of marginal workers is important for understanding labour market dynamics and socioeconomic conditions.

Details about the column used:

For this project, we are using certain columns such as Age group,District code,State code, Area Name, Worked for 3 months or more but less than 6 months - Persons, Worked for 3 months or more but less than 6 months – Males, Worked for 3 months or more but less than 6 months – Females, Worked for less than 3 months – Persons, Worked for less than 3 months – Males, Worked for less than 3 months – Females, Industrial Category - A - Cultivators – Persons, Industrial Category - A - Cultivators – Males, Industrial Category - A - Cultivators – Females, Industrial Category - A - Agricultural labourers – Persons, Industrial Category - A - Agricultural labourers – Males, Industrial Category - A - Agricultural labourers – Females, Industrial Category - A - Plantation, Livestock, Forestry, Fishing, Hunting and allied activities – Persons, Industrial Category - A - Plantation, Livestock, Forestry, Fishing, Hunting and allied activities – Males, Industrial Category - A - Plantation, Livestock, Forestry, Fishing, Hunting and allied activities – Females,etc.,

2. 5 DIFFERENT WAYS TO LOAD DATA IN PYTHON:

The ways that I am going to discuss are:

- Manual function
- loadtxt function
- genfromtxt function
- read_csv function
- Pickle

Reading Data:

Pandas: It's the most common library for reading and manipulating data in various formats, such as CSV, Excel, SQL databases, and more.

```
# Load your dataset (replace 'your_dataset.csv' with the actual dataset file path)

data = pd.read_csv('your_dataset.csv')
```

3. STEPS INVOLVED IN PREPROCESSING THE DATASET IN PYTHON:

Preprocessing a dataset in Python typically involves several steps to clean, transform, and prepare the data for machine learning or analysis. Here are the common steps involved:

- Data Import: Load the dataset into your Python environment. You can use libraries like Pandas to read data from various sources (CSV, Excel, SQL, etc.).
- Data Inspection: Examine the dataset to understand its structure, columns, and any missing values. You can use functions like head(), info(), and describe().
- Handling Missing Data: Deal with missing values by either removing rows or columns with missing data, or imputing missing values with mean, median, or mode values.
- Data Cleaning: Remove duplicates if present. Correct inconsistent data (e.g., typos, different representations of the same category).
- Data Transformation: Encode categorical variables using techniques like one-hot encoding or label encoding. Normalize or scale numerical features if necessary. Perform feature engineering to create new features based on domain knowledge.
- Data Splitting: Split the dataset into training and testing sets to evaluate the model's performance. You can use libraries like Scikit-learn to do this.
- Feature Selection: Choose relevant features based on their importance. You can use techniques like feature selection or dimensionality reduction.
- Data Visualization: Create visualizations to better understand the data and relationships between features. Libraries like Matplotlib and Seaborn can help with this.
- Outlier Detection: Identify and handle outliers if they exist in the dataset.
- Scaling and Standardization: Scale or standardize features to ensure they have similar scales, especially for algorithms sensitive to feature magnitudes.
- Data Splitting: Split the data into training and testing sets to assess the model's performance. Common ratios are 70-30 or 80-20.
- Data Preprocessing Pipelines: Create pipelines to automate the preprocessing steps. This is especially useful for consistency and reusability.
- Save Processed Data: Save the preprocessed data to a file for future use. Common formats include CSV or Pickle.

These steps may vary depending on your specific dataset and the problem you're trying to solve. It's essential to tailor the preprocessing steps to the needs of your project. Additionally, you can make use of various Python libraries like Pandas, NumPy, Scikit-learn, and Matplotlib to streamline the preprocessing process.

4. PERFORMING DIFFERENT ANALYSIS NEEDED:

Performing different analyses on a dataset in Python typically involves using various libraries and techniques. Here's a general process you can follow:

- Import Libraries: Start by importing the necessary libraries, such as Pandas for data manipulation, NumPy for numerical operations, Matplotlib or Seaborn for data visualization, and Scikit-Learn for machine learning tasks.

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split
```

- Load Your Dataset: Use Pandas to read your dataset into a DataFrame.

data = pd.read_csv('your_dataset.csv')

```
(eg.,) data=pd.read_csv("C:\\Users\\OOAD LAB\\Downloads\\DDW_B06ST_3300_State_TAMIL_NADU-2011.csv")
```

- Data Exploration:

Check the first few rows of your dataset with data.head().

Summary statistics with data.describe().

Data types and missing values with data.info().

```
(eg.,) data.info()
   data.describe()
   data.head(10)
   data.tail(10)
```

- Data Cleaning:

Handle missing data (e.g., using data.dropna() or data.fillna()).

Remove duplicates with data.drop_duplicates().

Data transformation if needed.

```
(eg.,) data.isna()
```

- Data Visualization:Create various plots to explore your data (e.g., histograms, scatter plots, box plots).Use libraries like Matplotlib and Seaborn for this.

```
(eg.,) import plotly.express as px

fig = px.histogram(mydataset,x="Worked for less than 3 months - Persons", title="marginal workers", color="Worked for less than 3 months - Persons")
# Update the layout and add box plots
fig.update_layout(
    bargap=0.2
)
fig.show()
```

- Data Analysis:Depending on your specific analysis goals, you can perform various tasks, including:

Statistical tests (e.g., t-tests, ANOVA).

Correlation analysis.

Regression analysis.

Classification or clustering if it's a machine learning problem.

- Machine Learning (if applicable): Split your data into training and testing sets using train_test_split(). Choose a machine learning algorithm that suits your problem. Train the model, make predictions, and evaluate its performance.

```
➢   from sklearn.model_selection import train_test_split
➢   from sklearn.linear_model import LinearRegression
```

```
from sklearn.metrics import mean_squared_error, r2_score
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20)
x_train.shape
```

- Data Presentation: Summarize your findings with clear visualizations and explanations. Use Jupyter Notebooks or other tools for creating reports.
- Iterate: Data analysis is often an iterative process. You might need to go back and make adjustments to your analysis based on the results.