ASSESSMENT OF MARGINAL WORKER IN TAMILNADU-A SOCIOECONOMIC ANALYSIS

PHASE 3: Development Part 1

TOPIC: Start building of socioeconomic analysis model by loading and pre-processing the dataset.

Introduction:

• Socioeconomic analysis is a multidisciplinary approach that examines the complex interplay between social and economic factors within a given society or community. It aims to understand and assess the impact of economic policies, practices, and disparities on the well-being and quality of life of individuals and groups.

• Socioeconomic analysis relies heavily on data collection, statistical analysis, and research. Researchers often use surveys, censuses, and other data sources to gather information that informs their analysis.

Dataset Overview:

Marginal workers dataset offers a comprehensive collection of data points, capturing a wide array of socioeconomic variables. This dataset encompasses factors such as income, education, employment, healthcare, demographic information, and more. By leveraging this dataset, we aim to gain valuable insights into the socioeconomic well-being and challenges faced by the target population.

Given dataset:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| District Code | Area Name | Total/  Rural  / Urban | .... | Industrial  Category - R  to U  - Non HHI –  Persons | Industrial  Category –  R to U  - Non HHI  - Males | Industrial Category - R to U - Non HHI - Females |
| `000 | State –  TAMIL NADU | Total | .... | 122088 | 55801 | 66287 |
| `000 | State –  TAMIL NADU | Total | .... | 19305 | 9774 | 9531 |
| `000 | State –  TAMIL NADU | Total | ... | 68929 | 32803 | 36126 |
| .  .  .  .  .  . | .  .  .  .  .  . | .  .  .  .  .  . | ....  ....  ....  ....  ....  .... | .  .  .  .  .  . | .  .  .  .  .  . | .  .  .  .  .  . |
| `633 | District - Tiruppur | Urban | .... | 279 | 103 | 176 |
| `633 | District - Tiruppur | Urban | .... | 81 | 35 | 46 |
| `633 | District - Tiruppur | Urban | .... | 0 | 0 | 0 |

595 rows x 69 columns

Necessary step to follow:

1.Import Libraries:

Start by importing the necessary libraries:

Program:

import numpy as np

import matplotlib.pyplot as plt

2. Load the dataset:

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

Program:

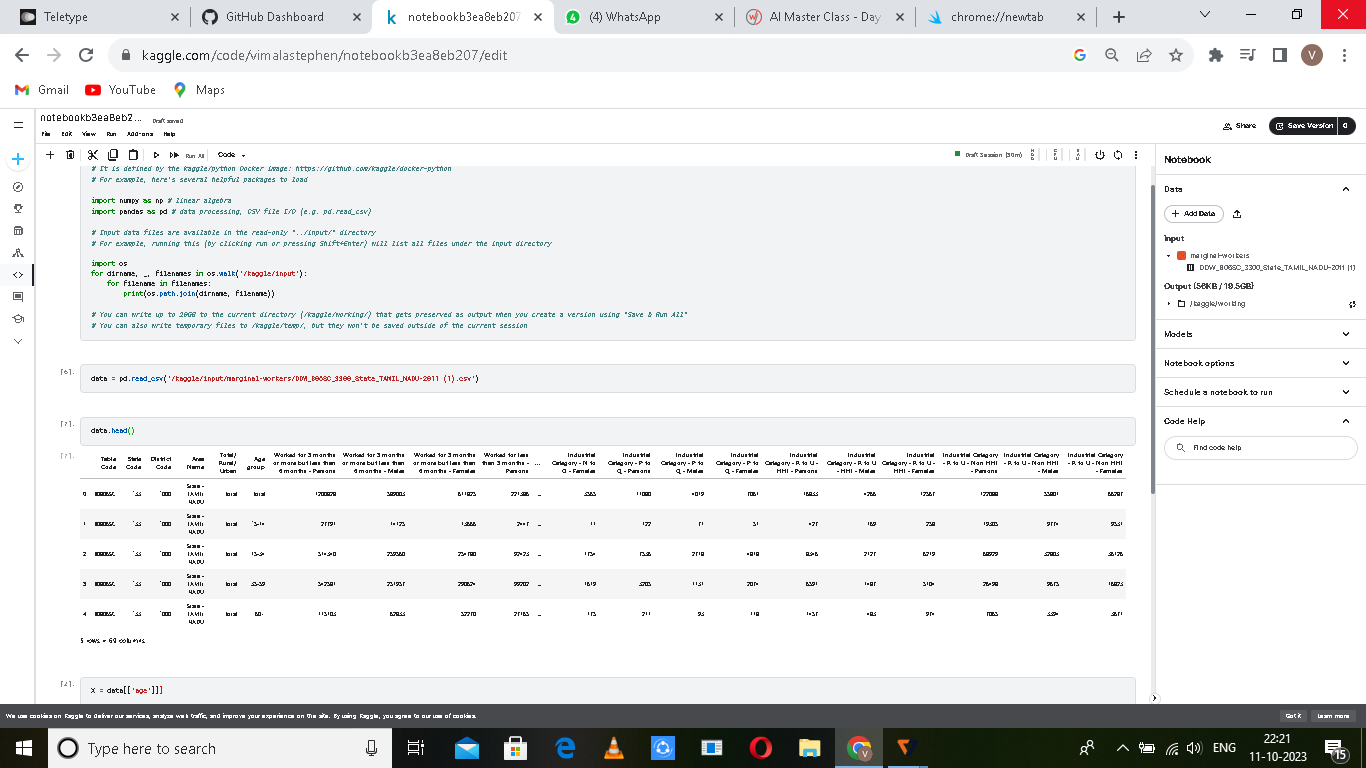
data = pd.read\_csv(' https://tn.data.gov.in/catalog/marginal-workers-classified-age-industrial-category-and-sex-census-2011-india-and-states.csv ')

Pd.read()

3. Explore the dataset by checking the first few rows to understand its structure:

data.head()

output as be like:



4. Pre-processing the dataset:

Pre-processing a dataset is a crucial step in data analysis and machine learning. It involves cleaning, transforming, and organizing the data to make it suitable for further analysis or modeling. Here are some common pre-processing steps:

Handling Missing Data:

Identify and handle missing values. You can choose to remove rows with missing data, fill in missing values using the mean or median, or employ more sophisticated imputation methods.

Data Cleaning: Remove duplicates: Check for and remove duplicate records from the dataset.

Outlier detection: Identify and handle outliers that may affect the analysis.

Data Transformation: Encoding categorical variables: Convert categorical data into a numerical format, for example, using one-hot encoding.

Scaling and normalization: Standardize numerical features to have a mean of 0 and a standard deviation of 1.

Feature Engineering: Create new features: Develop new variables that might provide more valuable insights. This can involve mathematical operations on existing features or domain-specific knowledge.

Data Reduction: Dimensionality reduction: If your dataset has many features, consider techniques like Principal Component Analysis (PCA) to reduce the dimensionality while retaining the most important information.

Data Splitting: Split the dataset into training and testing sets for model evaluation.

Here's a Python example using Pandas for some of these pre-processing steps:

import pandas as pd

from sklearn.model\_selection

import train\_test\_split

from sklearn.preprocessing

import StandardScaler

# Load the dataset

data = pd.read\_csv('' https://tn.data.gov.in/catalog/marginal-workers-classified-age-industrial-category-and-sex-census-2011-india-and-states.csv')

# Handling missing data

data.dropna(subset=['column\_with\_missing\_values'], inplace=True)

# OR fill missing values

data['column\_with\_missing\_values'].fillna(data['column\_with\_missing\_values'].mean(), inplace=True)

# Data transformation (e.g., one-hot encoding)

data = pd.get\_dummies(data, columns=['categorical\_column'])

# Scaling and normalization

scaler = StandardScaler()

data['numerical\_column'] = scaler.fit\_transform(data['numerical\_column'].values.reshape(-1, 1))

# Feature engineering

data['new\_feature'] = data['feature1'] \* data['feature2']

# Data splitting

X = data.drop('target\_variable', axis=1)

y = data['target\_variable']

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

5.Data manipulation libraries

Certainly, when working with data in Python, one of the most commonly used libraries for data manipulation is Pandas. Pandas is a powerful and versatile library that provides data structures and functions for working with structured data. Here's an overview of Pandas:

Pandas: Pandas is an open-source Python library that offers data structures and data analysis tools. It is particularly well-suited for data manipulation and analysis tasks. Here are some key features of Pandas:

DataFrame: Pandas introduces the DataFrame, a 2-dimensional, size-mutable, and heterogeneous tabular data structure. It is similar to a spreadsheet or a SQL table, making it an ideal choice for working with structured data.

Series: A Series is a one-dimensional array-like object that can hold data of various types. It is often used for representing a single column or row in a DataFrame.

Data Alignment: One of the strengths of Pandas is its ability to automatically align data along rows and columns. This makes it easy to perform operations on data that may not be perfectly aligned.

Data Cleaning: Pandas provides various functions for data cleaning, including handling missing data (NaN values), dropping duplicates, and filling in missing values.

Data Selection: You can use Pandas to select and filter data, create subsets, and perform advanced indexing and selection operations.

Data Transformation: It allows you to reshape data, pivot tables, and combine data from multiple sources. You can also apply functions and transformations to data.

Grouping and Aggregation: Pandas supports grouping data based on one or more keys and performing aggregation operations such as sum, mean, count, and more.

Time Series: It provides functionality for working with time series data, making it a valuable tool for financial and temporal data analysis.

Integration with Other Libraries: Pandas seamlessly integrates with other data analysis and visualization libraries like NumPy, Matplotlib, and Seaborn.

Here's a basic example of how to load a CSV file and perform some data manipulation using Pandas:

Import pandas as pd

# Load a dataset from a CSV file

Data = pd.read\_csv('https://tn.data.gov.in/catalog/marginal-workers-classified-age-industrial-category-and-sex-census-2011-india-and-states.csv')

# Display the first few rows of the dataset

print(data.head())

# Select a specific column

Age Group = data['Age Group']

# Group and aggregate data

grouped\_data = data.groupby('Category')['Value'].mean()

My Program:

import pandas as pd

import numpy as np

import plotly.express as px

import matplotlib.pyplot as plt

data = pd.read\_csv('DDW\_B06SC\_3300\_State\_TAMIL\_NADU-2011 (1).csv')

print(data.head())

print(data.describe())

print(data.isnull().sum())

plt.hist(data['Area Name'], bins=20)

plt.xlabel('areas')

plt.ylabel('workers in the areas')

plt.title('Histogram of Area Name')

plt.show()

x = data['Industrial Category - A - Cultivators - Males']

y = data['Industrial Category - A - Cultivators - Females']

plt.scatter(x, y, marker='o', color='blue', label='Scatter Plot')

plt.title('Scatterplot of Industrial Category - A - Cultivators - Males vs. Industrial Category - A – Cultivators - Females')

plt.xlabel('Industrial Category - A - Cultivators - Males')

plt.ylabel('Industrial Category - A - Cultivators - Females')

plt.legend()

plt.grid(True)

plt.show()

OUTPUT:

