ADS_PHASE5

ASSESSMENT OF MARGINLAL WORKERS IN TAMILNADU-A SOCIOECONOMIC ANALYSIS

1. Problem definition:

The primary goal of this project is to comprehensively assess the socioeconomic conditions and characteristics of marginal workers in Tamil Nadu, with a focus on gaining insights into their employment patterns, living conditions, and well-being. This assessment aims to address the following key questions:

- Definition of Marginal Workers: Clearly define the criteria and characteristics that categorize individuals as marginal workers in the context of Tamil Nadu.
- Population Size: Determine the size and distribution of the marginal worker population across different regions within Tamil Nadu.
- Employment Patterns: Analyze the types of employment and sectors in which marginal workers are engaged. Identify any seasonal variations or shifts in employment.
- Income and Earnings: Investigate the income levels, earnings, and wage disparities among marginal workers compared to other occupational categories.
- Living Conditions: Assess the housing conditions, access to basic amenities, and healthcare facilities available to marginal workers and their families.
- Social and Economic Inclusion: Examine the level of social and economic inclusion of marginal workers, including access to education, social security, and government welfare schemes.
- Challenges and Vulnerabilities: Identify the unique challenges, vulnerabilities, and risks faced by marginal workers, such as job insecurity, health issues, or discrimination.
- Policy Evaluation: Evaluate the effectiveness of existing government policies and programs targeted at improving the livelihoods and well-being of marginal workers.

2. Design thinking:

- Data collection
- Data preprocessing
- Feature engineering
- Model selection
- Model training
- Evaluation

➢ Data collection:

The process of gathering and analyzing accurate data from various sources to find answers to research problems, trends, and probabilities, etc., to evaluate possible outcomes is Known as Data collection.

Methods of data collection:

• Surveys, quizzes, and questionnaires.

• Interviews.

- Focus groups.
- Direct observation.

➢ Data preprocessing:

Data processing, manipulation of data by a computer. It includes the conversion of raw data to machine-readable form, flow of data through the CPU and memory to output devices, and formatting or transformation of output. Any use of computers to perform defined operations on data can be included under data processing.

The four main stages of data processing cycle are:

- Data collection.
- Data input.
- Data processing.
- Data output.

➢ Feature engineering:

Feature engineering involves a set of techniques that enable us to create new features by combining or transforming the existing ones. These techniques help to highlight the most important patterns and relationships in the data, which in turn helps the machine learning model to learn from the data more effectively.

Types of feature Engineering:

•7 of the Most Used Feature Engineering Techniques. Hands-on Feature Engineering with Scikit-Learn, TensorFlow, Pandas and Spicy.

•Encoding. Feature encoding is a process used to transform categorical data into numerical values that can be understood by ML algorithms.

•Feature Hashing.

•Binning / Bucketizing.

•Transformation.

➢ Model Selection:

✓ Time Series Forecasting

There are four components that a time series forecasting model is comprised of:

- Trend
- Seasonality
- Cyclical variations
- Random or irregular variations

✓ Autoregressive (AR)
✓ Autoregressive Integrated Moving Average (ARIMA)
✓ Seasonal Autoregressive Integrated Moving Average (SARIMA):

➢ Model training:

Model training is the phase in the data science development lifecycle where practitioners try to fit the best combination of weights and bias to a machine learning algorithm to minimize a loss function over the prediction range. The purpose of model training is to build the best mathematical representation of the relationship between data features and a target label (in supervised learning) or among the features themselves (unsupervised learning). Loss functions are a critical aspect of model training since they define how to optimize the machine learning algorithms. Depending on the objective, type of data and algorithm, data science practitioner use different type of loss functions. One of the popular examples of loss functions is Mean Square Error (MSE)

- Evaluation:

  Maybe the most popular and simple error metric is MAE: MAE: The Mean Absolute Error is defined as: While the MAE is easily interpretable (each residual contributes proportionally to the total amount of error), one could argue that using the sum of the residuals is not the best choice, as we could want to highlight especially whether the model incur in some large errors.

3. PHASES OF DEVELOPMENT:

The development phases for working with a dataset in Python typically involve the following steps:

- Data Collection:

Obtain the dataset from various sources such as files, databases, APIs, or web scraping.

Import necessary libraries for data handling and analysis (e.g., pandas, numpy).

- Data Loading:

Load the dataset into your Python environment using appropriate functions like pd.read_csv() for CSV files or other relevant functions for different file formats.

- Data Exploration

Get a basic understanding of your data by checking its structure, dimensions, and the first few rows using df.head().

Summarize key statistics with df.describe() and check data types with df.info().

Handle missing values, outliers, and data cleaning as needed.

- Data Preprocessing:

Perform preprocessing tasks like feature selection, data transformation, encoding categorical variables, and scaling numerical features.

- Data Visualization:

Create data visualizations using libraries like Matplotlib or Seaborn to gain insights and understand the data distribution.

- Data Analysis:

Analyze the dataset to answer specific questions or gain insights.

Apply statistical tests, correlations, and other analysis techniques.

- Model Development:

If you're working on a machine learning or statistical analysis task, split the data into training and testing sets.

Build, train, and evaluate your models using libraries like scikit-learn, TensorFlow, or PyTorch.

- Model Evaluation:

Assess the performance of your models with appropriate metrics, cross-validation, and hyperparameter tuning.

- Visualization of Results:

Visualize model results and insights from the analysis to communicate findings effectively.

- Conclusion and Reporting:

Summarize the findings, draw conclusions, and prepare reports or presentations as necessary.

- Deployment (if applicable):

If your project involves deploying a model or a data-driven application, take the necessary steps to deploy it.

- Documentation and Code Cleanup:

Document your work, clean up your code, and make it readable and maintainable for future reference.

These are the general phases, but the specific steps and their order may vary depending on the nature of your project and the dataset you're working with.

4. Dataset and its explanation:

Dataset are taken from the skillup:

(i.e.) 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 ofdata 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 andinformation about these workers, such as:

1. Age group
2. Worked for 3 months or more but less than 6 monthspersons
3. Worked for 3 months or more but less than 6 monthsmales

4. Worked for 3 monts or more but less than 6 monthsfemales
5. Worked for less than 3 months persons
6. Worked for less than 3 months males
7. Worked for less than 3 months femalesEtc.,

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 workersdataset can vary depending on the source and purpose of thedata collection.

5. 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 cleaningdata. 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 througha few steps before it is could be used for making predictions.

Steps involved in data preprocessing :

- Importing the required Libraries
- Importing the data set
- Handling the Missing Data.
- Encoding Categorical Data.
- Splitting the data set into test set and training set.
- 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 thisdataset: Data.csv

Every time we make a new model, we will require to importNumpy and Pandas. Numpy is a Library which contains Mathematical functions and is used for scientific computingwhile 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 canbe 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 thesklearn.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 beimputed. 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 alongcolumns 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. Anyvariable 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 encodethese 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 usethe 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 trainingour model called the **training set** and the other for testing
the performance of our model called the **test set**. The split isgenerally 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 thesame indices) ,
4. Y_test (test part of the dependent variables associated with the X test sets, and therefore also thesame indices).

We will assign to them the test_train_split, which takes theparameters — arrays (X and Y), test_size (Specifies the ratioin 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)
```

6. 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 standardizationor Z-score normalization is used. This is done by using

"StandardScaler" class of "sklearn.preprocessing".

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
```

7. How to train and test the dataset:

```
# Splitting data into training and testing sets
X = data.drop('target', axis=1)
y = data['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
# Training a model
model = RandomForestClassifier()
model.fit(X_train, y_train)
# Testing the model
y_pred = model.predict(X_test)
```

8. What metrics used for check the accuracy in dataset:

To calculate accuracy, you can use the accuracy_score function from Scikit-learn. It measures the proportion of correctly predicted instances out of the total instances in the dataset.

```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_true, y_pred)
```

9. Algorithm:

```
# Import necessary libraries
import pandas as pd
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
# Load your dataset (replace 'your_dataset.csv' with the actual dataset file path)
data = pd.read_csv('your_dataset.csv')
# Data Cleaning (if needed)
# Example: Handle missing values
data.dropna(inplace=True)
# Split the dataset into features (X) and target labels (y)
X = data.drop('target', axis=1)
y = data['target']
# Split the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Train a machine learning model (Random Forest in this example)
model = RandomForestClassifier()
model.fit(X_train, y_train)
# Test the model by making predictions on the test set
y_pred = model.predict(X_test)
# Calculate accuracy to evaluate model performance
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
# Data Visualization (example)
# Create a scatter plot of two features
plt.scatter(data['feature1'], data['feature2'])
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Scatter Plot of Features')
plt.show()
```

10. Explanation about the choice of machine learning algorithm:

The choice of a machine learning algorithm depends on several factors, including the nature of the problem you're trying to solve, the available data, and your specific goals. Here are some considerations to help you make the right choice:

Type of Problem:

Classification: If you're dealing with problems where you need to categorize data into classes, algorithms like Logistic Regression, Decision Trees, Random Forest, Support Vector Machines (SVM), or deep learning models like Convolutional Neural Networks (CNN) for image classification might be suitable.

Regression: For problems involving predicting numerical values, linear regression, decision trees, or gradient boosting methods like XGBoost can be effective.

Data Size: For small datasets, simple models like linear regression or k-nearest neighbors might work well. For larger datasets, deep learning models or ensemble methods like Random Forest and Gradient Boosting can be more powerful.

Data Quality: If your data is noisy or contains missing values, some algorithms are more robust to handle these issues. Decision trees and random forests are examples of such algorithms.

Interpretability: If interpretability is crucial, linear models or decision trees are more transparent compared to complex models like neural networks.

Computational Resources: Deep learning models require significant computational resources, so consider the available hardware when choosing your algorithm.

Feature Space: If you have a high-dimensional feature space, dimensionality reduction techniques like Principal Component Analysis (PCA) can be used in conjunction with various algorithms.

Time Sensitivity: If you need real-time predictions, simpler models like linear regression or decision trees may be preferred due to their lower inference times.

Ensemble Methods: Ensemble methods like Random Forest, Gradient Boosting, and AdaBoost can often improve predictive performance by combining multiple models.

Domain Knowledge:Your understanding of the domain and problem can help guide your choice. Some algorithms might be more suitable for specific domains.

Experimentation: Don't be afraid to experiment with different algorithms and compare their performance using techniques like cross-validation.

Bias and Fairness: Be aware of bias and fairness issues in your data and consider algorithms that address these concerns, or implement pre-processing techniques to mitigate bias.

Scalability: Consider the scalability of the algorithm. If you plan to scale your solution, distributed frameworks like Apache Spark can be helpful.

Library and Framework Support: The availability of libraries and frameworks for a specific algorithm can affect your choice, as it can streamline development and deployment.