Assessment of marginal workers in tamilnadu -A socioeconomic analysis

Problem Statement:

Problem Statement: "There is a high rate of unemployment among young adults in a specific region, and it is negatively impacting the local economy and community well-being. The goal is to create a solution that helps improve employment opportunities and the economic situation in the region."

Design Thinking Process:

1. Empathize:

- Understand the experiences and needs of young adults in the region who are facing unemployment.

- Gather insights through surveys, interviews, and observations to empathize with their challenges and aspirations.

2. Define:

- Define the specific problems and constraints faced by the target population, such as limited access to education, lack of skills, and limited job opportunities.

- Create a clear problem statement and identify the key issues that need to be addressed.

3. Ideate:

- Brainstorm and generate creative ideas and solutions to tackle the unemployment problem.

- Encourage collaboration and diverse perspectives to come up with innovative approaches.

4. Prototype:

- Develop prototypes or small-scale solutions that address different aspects of the problem.

- These prototypes can be in the form of pilot programs, workshops, or initiatives to test the feasibility and effectiveness of the ideas.

5. Test:

- Implement the prototypes and gather feedback from the target audience.

- Evaluate the impact of the solutions and collect data on their effectiveness.

6. Iterate:

- Based on the feedback and data collected during the testing phase, refine and improve the solutions.

- Continuously iterate and make necessary adjustments to enhance the effectiveness of the interventions.

Phases of Development:

1. Research and Data Collection:

- Conduct research to gather data on the unemployment rate, demographics, and challenges faced by young adults in the region.

- Analyze existing programs and policies related to employment and education.

2. Needs Assessment:

- Identify the specific needs of the target population, such as skill development, access to job opportunities, and career counseling.

3. Solution Design:

- Based on the findings from the research and needs assessment, design a comprehensive solution that addresses the identified challenges.

- Develop a strategy that includes educational and vocational training programs, job placement services, and community engagement initiatives.

4. Implementation:

- Roll out the designed solutions, including launching training programs, creating partnerships with local businesses, and establishing support services.

- Monitor the progress and effectiveness of the initiatives.

5. Monitoring and Evaluation:

- Continuously collect data and monitor the impact of the implemented solutions.

- Assess the success of the interventions in reducing unemployment and improving economic well-being

6. Scaling and Sustainability:

- If successful, consider scaling up the initiatives to reach a broader audience.

- Develop long-term sustainability plans, which may include securing funding, establishing partnerships, and ensuring the continued success of the programs.

7. Community Engagement:

- Involve the local community and stakeholders in the development and implementation phases.

- Encourage participation, feedback, and collaboration to ensure that the solutions align with community needs and values.

Certainly, I can provide a general overview of the dataset, data preprocessing steps, and the model training process in the context of a typical machine learning project.

Dataset:

Here's a brief description of some of the key columns:

1.Table Code: A code or identifier for the data table.

2.State Code:A code representing the state (Tamil Nadu in this case)

3.District Code:A code representing the specific district within Tamil Nadu

4.Area Name: The name or identifier of the geographic area being analyzed, which could be a specific district, town, or region.

5.Total/Rural/Urban: This column may indicate whether the data relates to the total population, rural areas, or urban areas within the specified geographic region.

6.Age Group: Age group categories for the population being analyzed.

7.Worked for 3 months or more but less than 6 months - Persons/Males/Females: These columns likely represent the number of individuals, males, and females who have worked for a specific duration (3 to 6 months).

8.Worked for less than 3 months - Persons/Males/Females: These columns likely represent the number of individuals, males, and females who have worked for a shorter duration (less than 3 months).

9.Industrial Categories A to U - Persons/Males/Females: These columns likely represent the distribution of individuals across various industrial categories, such as cultivators, agricultural laborers, plantation workers, and other occupational categories. The data appears to be segregated by gender (males and females).

10.HHI (Household Industries) vs. Non HHI: These categories might be related to whether individuals are employed in household industries (HHI) or non-household industries (non-HHI).

11.Category Letters (A, B, C, etc.): These letters represent different industrial categories or sectors, each with specific codes or identifiers.

Data Preprocessing Steps:

Data preprocessing is a critical step to ensure the data is clean and suitable for model training:

Data Cleaning:

- Remove duplicate records.

- Handle missing values, either by imputing them or removing rows with missing values.

- Address outliers if they exist.

Text Data Processing:

- Tokenization: Split the review text into individual words or tokens.

- Stopword Removal: Remove common words (e.g., "the," "and," "is") that do not carry significant meaning.

- Text Lowercasing: Convert all text to lowercase to ensure consistency.

- Text Lemmatization or Stemming: Reduce words to their root form to normalize the text.

Feature Engineering:

- Create new features if necessary, like word count, average word length, or sentiment scores based on the review text.

Encoding Categorical Variables:

- Convert categorical variables (e.g., app version) into numerical representations using techniques like one-hot encoding or label encoding.

Train-Test Split:

- Split the dataset into training and testing sets to evaluate model performance. Common splits are 70-30 or 80-20.

Model Training Process:

After data preprocessing, you can move on to the model training process:

Feature Selection:

- Choose relevant features for training the model. This may involve selecting the most informative features using techniques like feature importance or correlation analysis.

Model Selection:

- Select an appropriate machine learning or deep learning model for your task. In this case, you might choose a text classification model like a Naive Bayes classifier, a support vector machine, or a deep learning model like a neural network.

Training the Model:

- Split the training dataset further into a training set and a validation set for hyperparameter tuning.

- Train the model on the training data and validate its performance on the validation set.

Hyperparameter Tuning:

- Optimize model hyperparameters to improve model performance. Techniques like grid search or random search can be used.

Model Evaluation:

- Evaluate the model's performance on the testing dataset using appropriate metrics.

Model Deployment:

- If the model performs satisfactorily, deploy it in a real-world application where it can make predictions on new data.

Monitoring and Maintenance:

- Continuously monitor the model's performance in a production environment and retrain it as needed with new data.

The choice of a time series forecasting algorithm and evaluation metrics depends on the characteristics of the time series data and the specific forecasting objectives. Here's an explanation of how to choose an appropriate algorithm and evaluation metrics:

Choice of Time Series Forecasting Algorithm:

ARIMA (AutoRegressive Integrated Moving Average):

-Suitable for stationary time series data, where patterns repeat over time. ARIMA models account for auto-correlation, seasonality, and trends.

Evaluation Metrics:

1. Mean Absolute Error (MAE):

- MAE measures the average absolute difference between the predicted values and the actual values. It gives equal weight to all errors.

2.Mean Squared Error (MSE):

- MSE measures the average squared difference between predictions and actual values. It penalizes larger errors more than MAE.

3.Root Mean Squared Error (RMSE):

- RMSE is the square root of the MSE. It is in the same unit as the target variable, making it more interpretable.

4. Mean Absolute Percentage Error (MAPE):

- MAPE expresses errors as a percentage of the actual values. It is particularly useful when you want to understand the forecasting error relative to the scale of the data.

5.Custom Domain-Specific Metrics:

- Depending on the application, you may define custom metrics. For instance, if you're forecasting sales, you might create a metric that considers inventory levels and supply chain disruptions.

Code:

import pandas as pd

import numpy as np

from statsmodels.tsa.holtwinters import ExponentialSmoothing

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

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

train\_size = int(len(data) \* 0.8)

train\_data, test\_data = data[:train\_size], data[train\_size:]

model = ExponentialSmoothing(train\_data['value'], trend='add', seasonal='add', seasonal\_periods=12)

model\_fit = model.fit()

forecast = model\_fit.forecast(len(test\_data))

mae = mean\_absolute\_error(test\_data['value'], forecast)

mse = mean\_squared\_error(test\_data['value'], forecast)

rmse = np.sqrt(mse)

r2 = r2\_score(test\_data['value'], forecast)

plt.figure(figsize=(12, 6))

plt.plot(train\_data, label='Training Data')

plt.plot(test\_data, label='Test Data')

plt.plot(forecast, label='Forecast')

plt.legend()

plt.title('Time Series Forecasting')

plt.show()

print(f'Mean Absolute Error: {mae}')

print(f'Mean Squared Error: {mse}')

print(f'Root Mean Squared Error: {rmse}')

print(f'R-squared (R2) Score: {r2}')