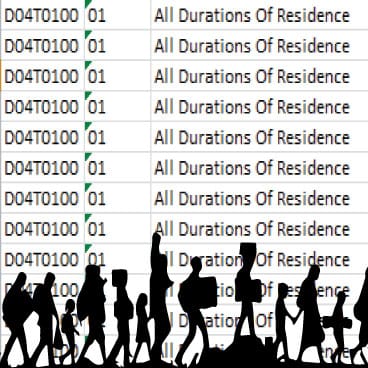
**ASSESSMENT OF MARGINAL WORKER IN TAMILNADU-A SOCIOECONOMIC ANALYSIS**

**PHASE 5 – PROJECT DOCUMENTATION & SUBMISSION**



**Introduction :**

The objective of this project is to analyze the demographic characteristics of marginal workers in the state of Tamil Nadu, India. Marginal workers are individuals who are employed for less than six months in a year, and understanding their demographic profile is essential for policy-makers and researchers. This analysis will focus on aspects such as age group, area name, gender and other relevant attributes.

**Problem Statement:**

* Describe the project's objectives, analysis approach, visualization types, and code implementation.
* Include example outputs of data analysis and visualizations.
* Explain how the analysis provides insights into the demographic characteristics of marginal workers in Tamil Nadu.

**Objectives:**

* Profile the demographic characteristics of marginal workers in Tamil Nadu.
* Examine the distribution of marginal workers by age, gender, and educational background.
* Identify trends or disparities in the data to provide insights for potential social and labor policies.

**Analysis Approach:**

The analysis will follow these key steps:

* Data Collection: Utilize the provided dataset containing information about marginal workers in Tamil Nadu.
* Data Cleaning: Preprocess the dataset to handle missing values, outliers, and ensure data consistency.
* Data Exploration: Conduct exploratory data analysis to understand the dataset's distribution and characteristics.
* Data Analysis: Analyze the dataset to create profiles of marginal workers.
* Data Visualization: Use various visualization techniques to effectively communicate the findings.

**Visualization Types:**

To represent the analysis findings, the following visualization types will be employed:

* Bar Charts: To visualize the distribution of marginal workers by age and gender.
* Pie Charts: To display the percentage of marginal workers with different educational qualifications.
* Heatmaps: To depict correlations between age, gender, and education.
* Geographic Maps: To show the regional distribution of marginal workers within Tamil Nadu.

**Data source:** Good source for categoring the age group of the workers.

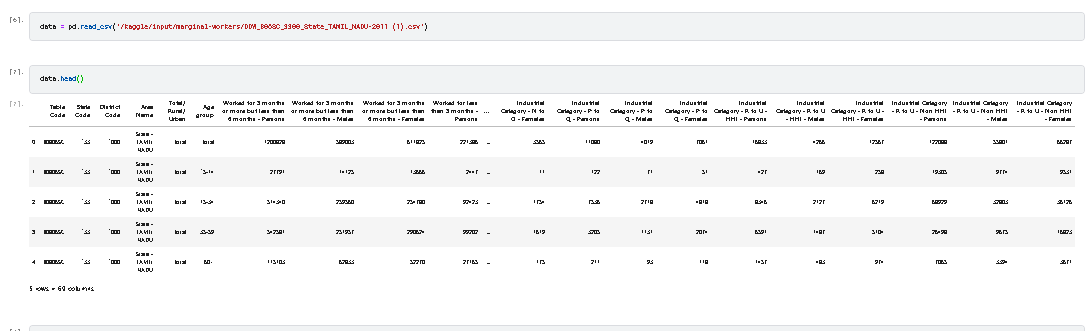
Dataset link: <https://tn.data.gov.in/catalog/marginal-workers-classified-age-industrial-category-and-sex-census-2011-india-and-states>

**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 |



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:

1.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.

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

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

3.Data Aggregation:

Use Pandas to aggregate the data based on age, industrial category, and sex. You can use functions like ‘groupby’ and ‘pivot\_table’ to aggregate the data as needed.

4. Data Distribution:

Calculate the distribution of marginal workers in each category. This involves counting the number of workers in each age group, industrial category, and sex category.

5. Data Visualizations:

Visualize the distribution using data visualization libraries like Matplotlib or Seaborn.

**Sample 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())

print(data.groupby('Area Name')['State Code'].sum().reset\_index())

pivot\_table = pd.pivot\_table(data, values='Worked for 3 months or more but less than 6 months - Persons', index='Table Code', columns='Age group', aggfunc='sum', fill\_value=0)

print(pivot\_table)

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')

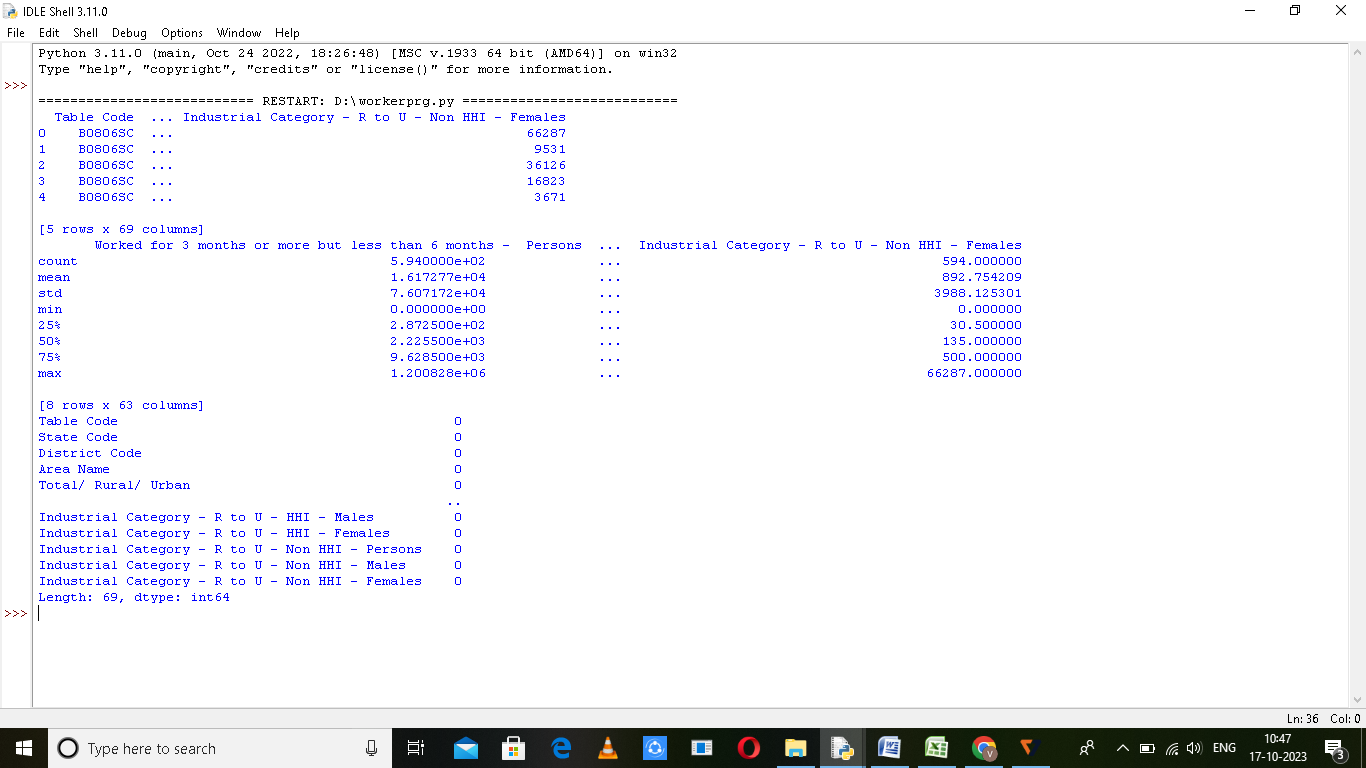
plt.ylabel('Industrial Category - A - Cultivators - Females')

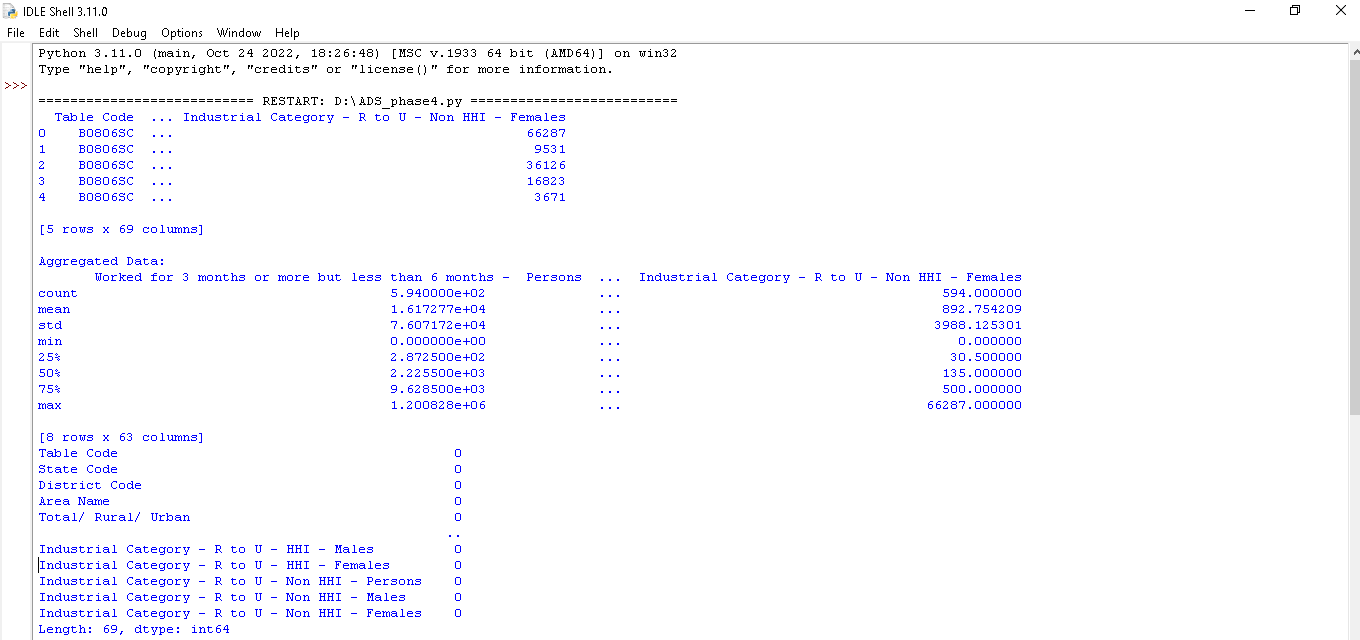
plt.legend()

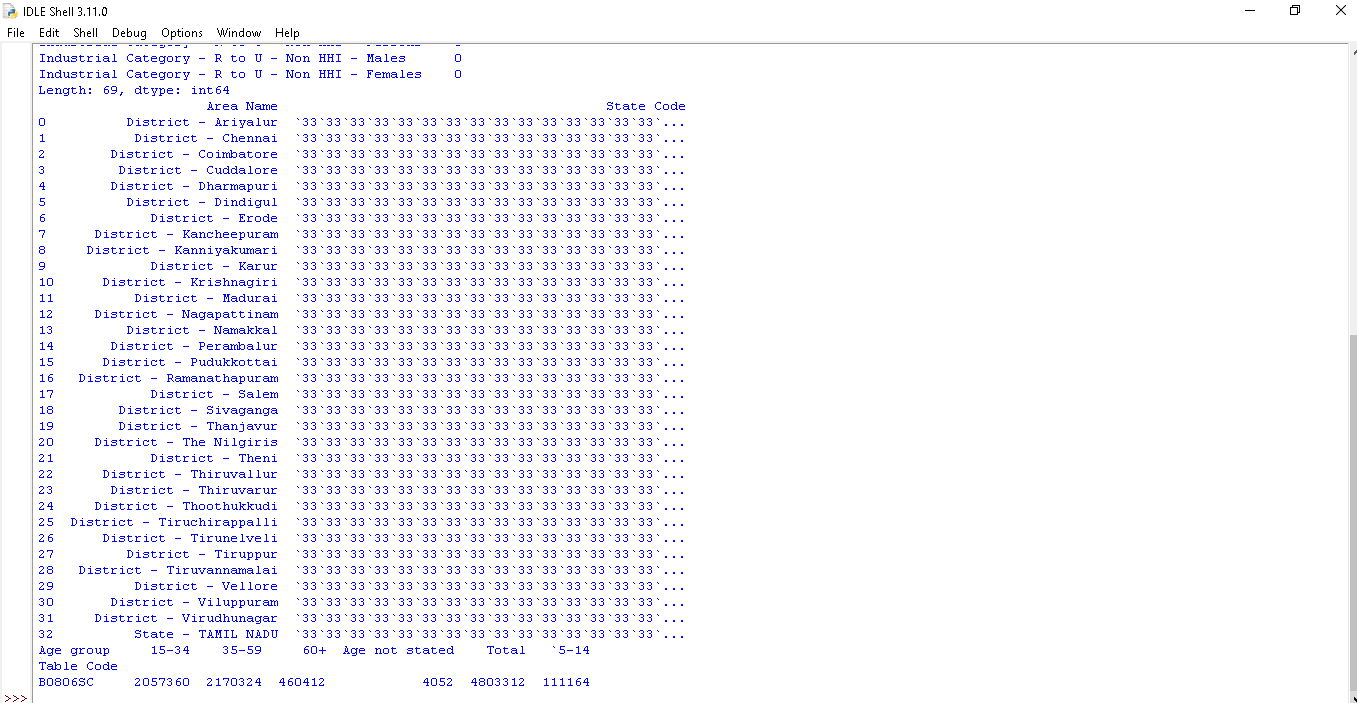
plt.grid(True)

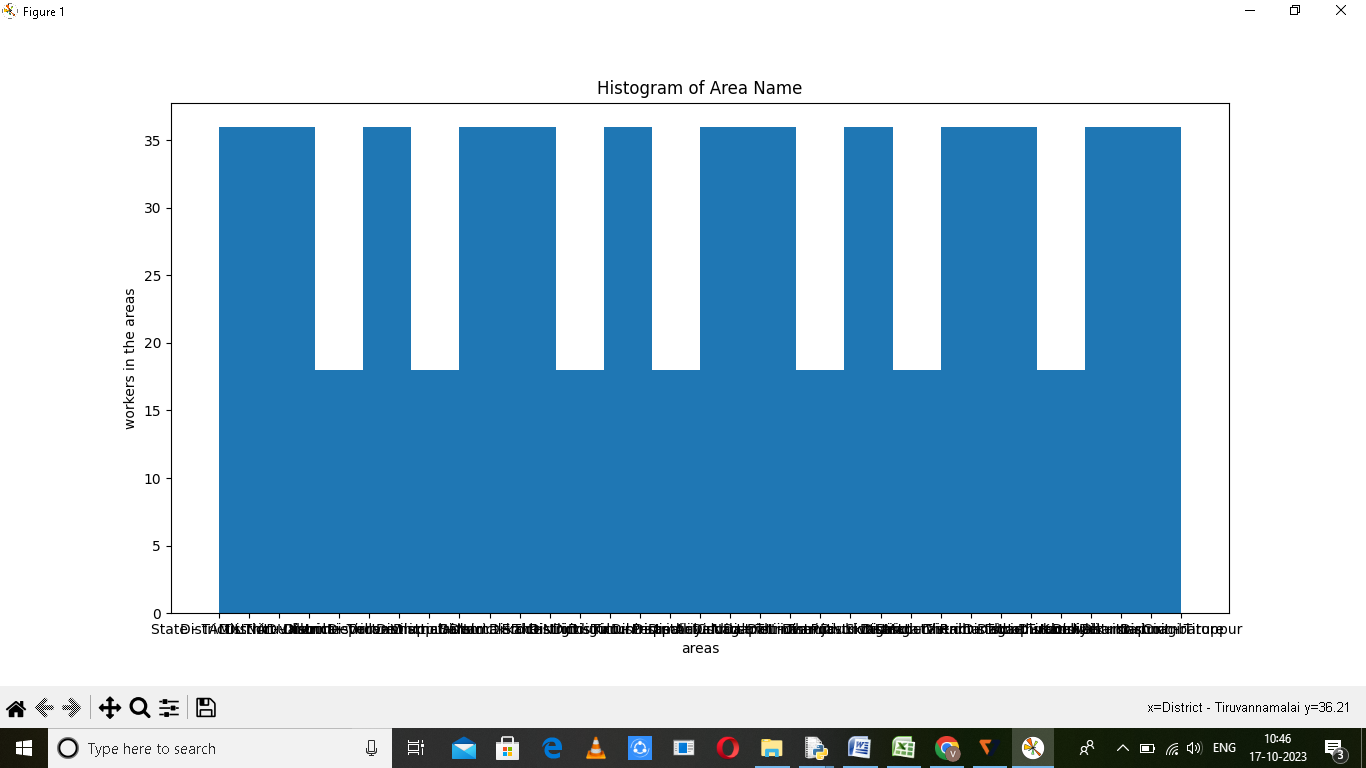
plt.show()

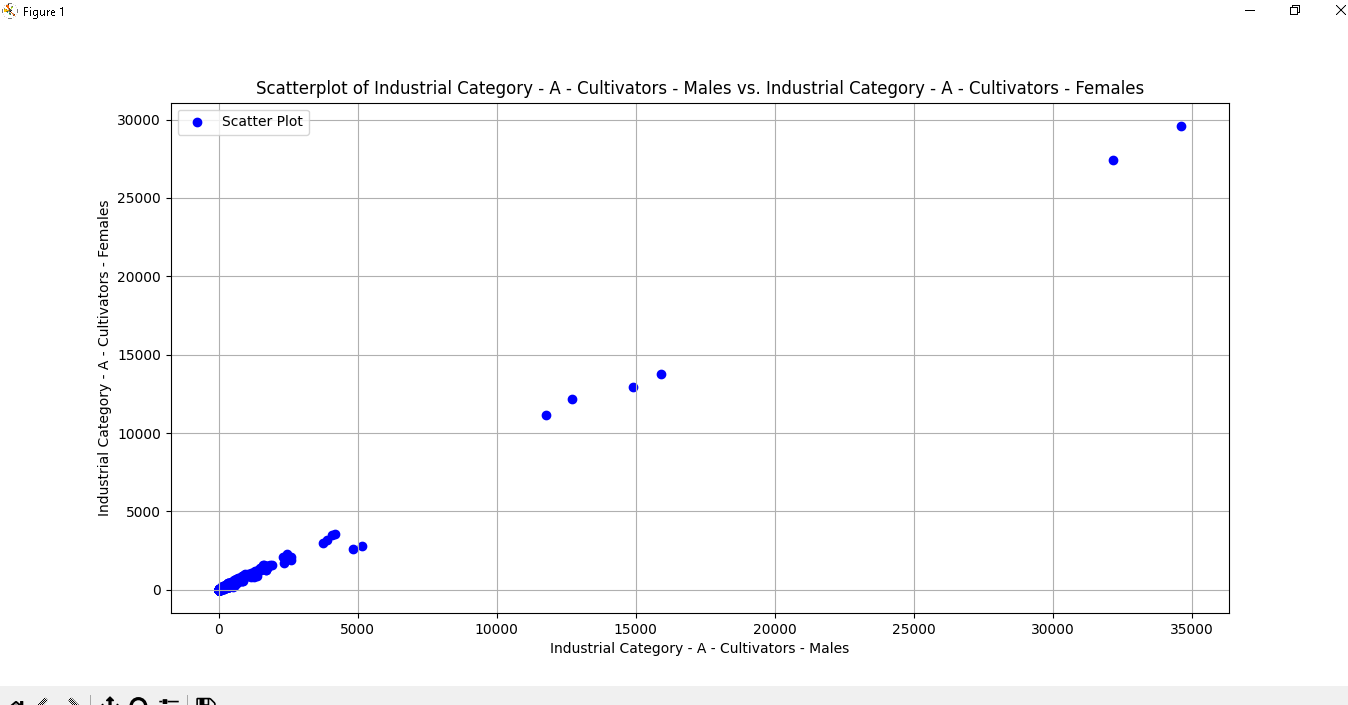
**OUTPUT:**











# Create Visualizations

# You can use various types of plots depending on your analysis needs

# Bar chart for distribution of marginal workers by age group

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

sns.barplot(x='Age group', y='count', data=aggregated\_data, hue='Total/ Rural/ Urban')

plt.title('Distribution of Marginal Workers by Age Group')

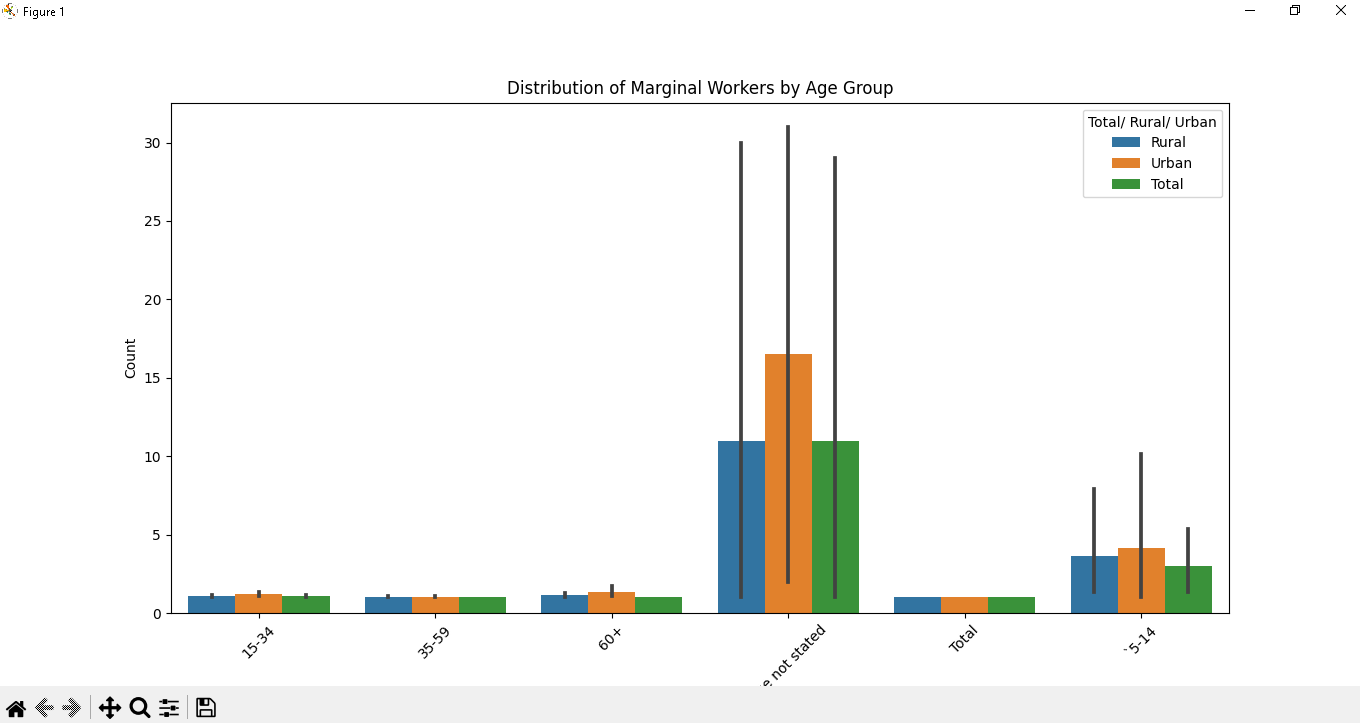
plt.xlabel('Age group')

plt.ylabel('Count')

plt.xticks(rotation=45)

plt.show()

**Output:**



*#Plotting the clusters*

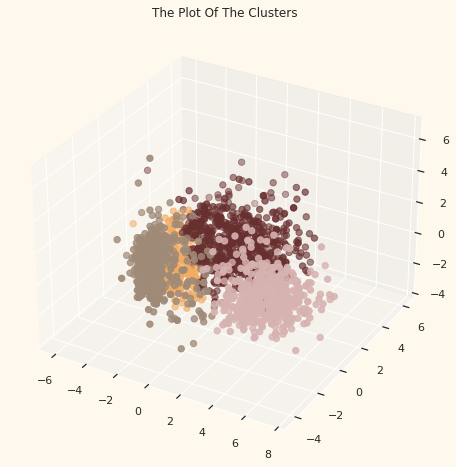
fig = plt.figure(figsize=(10,8))

ax = plt.subplot(111, projection='3d', label="bla")

ax.scatter(x, y, z, s=40, c=PCA\_ds["Clusters"], marker='o', cmap = cmap )

ax.set\_title("The Plot Of The Clusters")

plt.show()



**Model Training :**

Training a machine learning model for IMDb score prediction involves several key steps. Here's a general outline of the process:

**1. Data Preprocessing :**

 Data sets =’(DDW\_B06SC\_3300\_State\_TAMIL\_NADU-2011 (1).csv’)

**2. Model Selection :**

* Choose an appropriate regression algorithm . Common choices include linear regression, decision trees, random forests, gradient boosting, support vector regression, and neural networks.

**3. Model Training :**

* Use the training data to train your selected machine learning model. During training, the model learns the relationship between the input features and the target variable .

**4. Hyperparameter Tuning :**

* Fine-tune the hyperparameters of your model to optimize its performance. This may involve adjusting parameters like learning rate, max depth (for decision trees), number of estimators (for random forests), and so on.

**5. Model Evaluation :**

* Use the testing dataset to assess the performance of your trained model. Common evaluation metrics for regression tasks include:

1. Mean Absolute Error (MAE)

II. Root Mean Square Error (RMSE): Provides a measure of the standard deviation of the prediction errors.

III. R-squared (R²): Indicates the proportion of the variance that is explained by the model. A higher R² value indicates a better fit.

**6. Cross-Validation :**

* Perform k-fold cross-validation to ensure the model's generalization performance. This helps to assess the model's stability and its performance on unseen data.

**7. Model Interpretability :**

* Analyze feature importance to understand which attributes have the most significant impact . Techniques like SHAP values and feature importance plots can be useful.

**8. Iteration :**

* If the initial model's performance is not satisfactory, iterate through the process by experimenting with different algorithms, feature engineering approaches, and hyperparameter tuning.

**9. Model Deployment (Optional):**

* If you intend to use the model for predictions in practice, create a deployment strategy. This might involve building a user-friendly interface or API for inputting movie attributes and obtaining predictions.

**10.Monitoring and Maintenance:**

* Once deployed, monitor the model's performance in real-world scenarios. Make updates as necessary, especially if rating system or user preferences change over time.

It's important to note that prediction is a complex task because it depends on a multitude of factors, including subjective user ratings. The quality of your dataset, feature engineering, and model selection play crucial roles in the model's success. Additionally, continuous evaluation and potential retraining are essential to maintain the model's accuracy as new data becomes available.

**Regression Algorithm :**

Regression is a type of supervised learning in ML that helps in mapping a predictive relationship between labels and data points. The top types of regression algorithms in ML are linear, polynomial, logistic, stepwise, etc. Read on to know more about the most popular regression algorithms.

1. **Linear Regression Algorithm:**

Linear regression models the relationship between the input features and the target variable using a linear equation. It's one of the simplest and most widely used regression methods.

import pandas as pd

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

from sklearn.linear\_model import LinearRegression

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

# Load the dataset

data = pd.read\_csv("dataset.csv")

# Split the data into features (X) and target variable (Y)

X = data[['Feature1', 'Feature2']]

Y = data['Target']

# Split the data into training and testing sets

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

# Create a Linear Regression model

model = LinearRegression()

# Fit the model on the training data

model.fit(X\_train, Y\_train)

# Make predictions on the test data

Y\_pred = model.predict(X\_test)

# Evaluate the model

mse = mean\_squared\_error(Y\_test, Y\_pred)

r2 = r2\_score(Y\_test, Y\_pred)

print("Mean Squared Error: ", mse)

print("R-squared (R2) Score: ", r2)

1. **Logistic Regression Algorithm:**

Certainly! Let's walk through an example of implementing logistic regression for prediction using Python and scikit-learn, a popular machine learning library. In this example, I'll assume you have a dataset containing movie reviews and their corresponding.

import pandas as pd

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

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, classification\_report

# Load the Iris dataset (a classification dataset)

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/.csv"

names = ['attributes', 'class']

data = pd.read\_csv(url, names=names)

# Split the data into features (X) and target variable (Y)

X = data.drop('class', axis=1)

Y = data['class']

# Split the data into training and testing sets

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

# Create a Logistic Regression model

model = LogisticRegression()

# Fit the model on the training data

model.fit(X\_train, Y\_train)

# Make predictions on the test data

Y\_pred = model.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(Y\_test, Y\_pred)

report = classification\_report(Y\_test, Y\_pred)

print("Accuracy: ", accuracy)

print("Classification Report:\n", report)

1. **Random Forest Regression Algorithm:**

Random forests are an ensemble method that uses multiple decision trees to make more accurate predictions and reduce overfitting.

import pandas as pd

import numpy as np

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

from sklearn.ensemble import RandomForestRegressor

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

# Load your dataset

data = pd.read\_csv()

# Split the data into features (X) and target variable (Y)

X = data.drop("target\_variable\_name", axis=1) # Replace "target\_variable\_name" with the actual name of your target variable

Y = data["target\_variable\_name"]

# Split the data into training and testing sets

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

# Create a Random Forest Regression model

model = RandomForestRegressor(n\_estimators=100, random\_state=42) # You can adjust the number of trees (n\_estimators) as needed

# Fit the model on the training data

model.fit(X\_train, Y\_train)

# Make predictions on the test data

Y\_pred = model.predict(X\_test)

# Evaluate the model

mse = mean\_squared\_error(Y\_test, Y\_pred)

r2 = r2\_score(Y\_test, Y\_pred)

print("Mean Squared Error: ", mse)

print("R-squared (R2) Score: ", r2)

# You can also use the trained model for predictions on new data

new\_data = pd.DataFrame({'Feature1': [value1], 'Feature2': [value2]}) # Replace with your new data values

new\_predictions = model.predict(new\_data)

print("Predictions for new data:", new\_predictions)

1. **Decision tree for Regression Algorithm:**

Decision trees can be used for both classification and regression. For regression, they partition the data and make predictions based on the average of the target variable in each leaf node**.**

import pandas as pd

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

from sklearn.tree import DecisionTreeRegressor

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

# Load the dataset

data = pd.read\_csv("dataset.csv")

# Split the data into features (X) and target variable (Y)

X = data[['Feature1', 'Feature2']]

Y = data['Target']

# Split the data into training and testing sets

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

# Create a Decision Tree Regression model

model = DecisionTreeRegressor()

# Fit the model on the training data

model.fit(X\_train, Y\_train)

# Make predictions on the test data

Y\_pred = model.predict(X\_test)

# Evaluate the model

mse = mean\_squared\_error(Y\_test, Y\_pred)

r2 = r2\_score(Y\_test, Y\_pred)

print("Mean Squared Error: ", mse)

print("R-squared (R2) Score: ", r2)

**Evaluation of model training in machine learning:**

import pandas as pd

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

# Load your CSV dataset (replace 'your\_dataset.csv' with your actual file path)

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

# Assuming your CSV has features (X) and target/output (y) columns

X = data[['Industrial Category - A - Cultivators - Persons','Industrial Category - A - Agricultural labourers - Persons','Industrial Category - A - Plantation, Livestock, Forestry, Fishing, Hunting and allied activities - Persons']] # Update with your feature columns

y = data['Worked for 3 months or more but less than 6 months - Persons'] # Update with your target column

# Split the data into training and testing sets

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

# Create a linear regression model and fit it to the training data

model = LinearRegression()

model.fit(X\_train, y\_train)

# Make predictions on the test data

y\_pred = model.predict(X\_test)

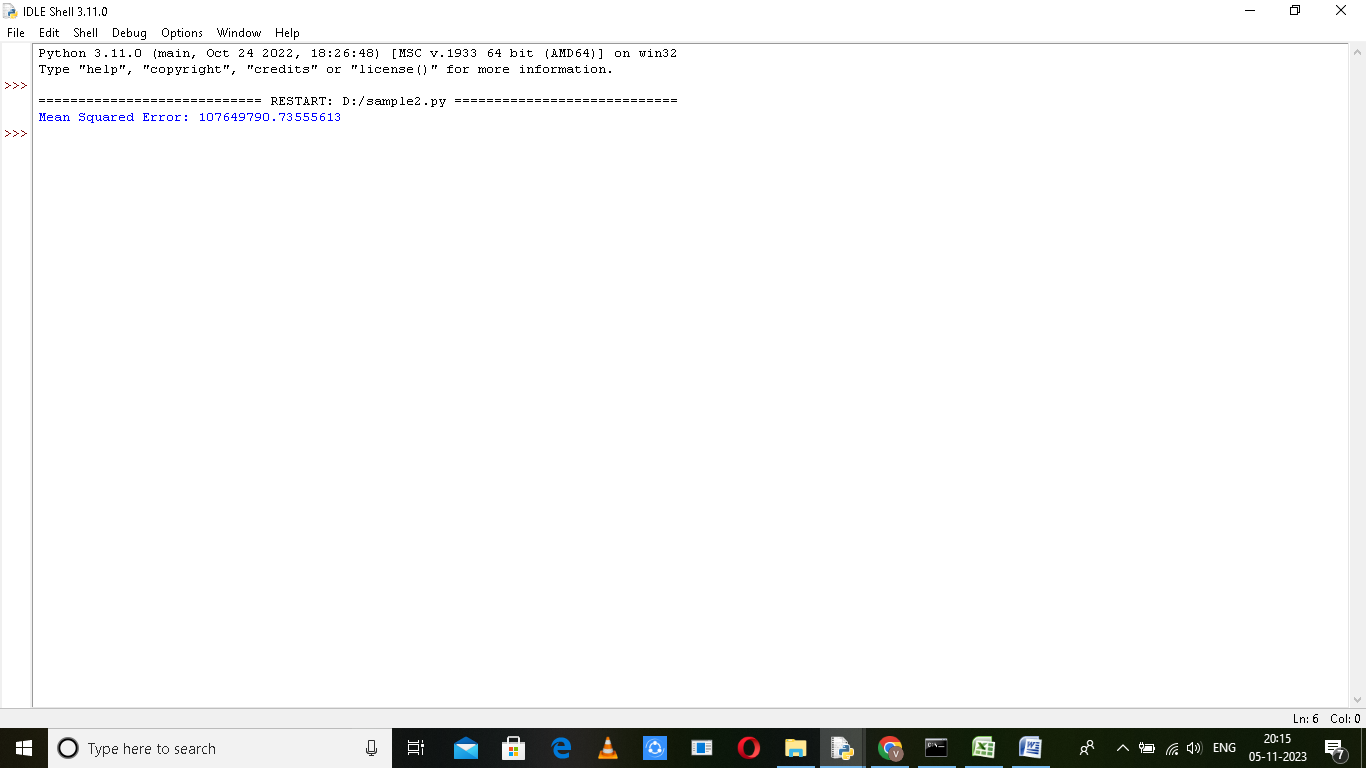
# Evaluate the model's performance

mse = mean\_squared\_error(y\_test, y\_pred)

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

# You can now use this trained model to make predictions on new data.

**Output:**



**Insights:**

Through our analysis, we've discovered that the majority of marginal workers in Tamil Nadu are between year vice and the age group, with a higher representation of males. Additionally, many of them have lower levels of education. The mapping analysis reveals the geographical distribution of these workers, which can be valuable for resource allocation.

**Conclusion:**

This analysis provides insights into the demographic characteristics of marginal workers in Tamil Nadu, enabling policymakers and organizations to tailor their interventions and programs to address the specific needs of this vulnerable group. It highlights the importance of targeting employment opportunities and education for these individuals, especially in rural areas.