MARGINAL WORKERS TAMILNADU

INTRODUCTION

Marginal workers are individuals in the labor force who have limited or irregular employment, often characterized by low wages, unstable job prospects, and generally precarious working conditions. They are typically at the fringes of the formal labor market and face challenges in securing stable and decent employment. Here are some key characteristics and aspects of marginal workers.

SITUATIONS FACED BY MARGINAL WORKERS

1. **Low Income**: Marginal workers typically earn very low wages, often below the minimum wage, and may struggle to make ends meet. Their low income can lead to poverty and economic insecurity.
2. **Job Insecurity**: These workers often have unstable and temporary employment, which can include part-time, contract, or casual work. This lack of job security can make it difficult for them to plan for the future and maintain a stable lifestyle.
3. **Lack of Benefits**: Marginal workers often lack access to employee benefits, such as health insurance, retirement plans, paid leave, and other forms of social security. This leaves them vulnerable to financial shocks and health crises.
4. **Limited Education and Training**: Many marginal workers have limited access to education and vocational training, which can restrict their ability to secure better-paying and more stable employment. This lack of skill development can perpetuate their economic disadvantage.
5. **Gender and Minority Disparities**: Marginal workers can disproportionately include women, minorities, and other disadvantaged groups who face additional challenges and discrimination in the labor market.
6. **Informal or Unregulated Work**: They may work in the informal or unregulated sector of the economy, which often lacks labor protections, including minimum wage laws and workplace safety regulations. This exposes them to exploitation and unsafe working conditions.
7. **Geographic Mobility**: Some marginal workers may be migrants who move to different regions or countries in search of employment. They may face challenges related to displacement, social integration, and discrimination.
8. **Vulnerability to Economic Shocks**: Marginal workers are more susceptible to economic downturns, such as recessions, as they may not have savings or job security to withstand job losses or reduced hours.
9. **Lack of Union Representation**: Many marginal workers are not unionized or have limited collective bargaining power, which can further hinder their ability to negotiate for better wages and working conditions.
10. **Social Exclusion**: Marginal workers may experience social exclusion and stigma due to their economic circumstances, which can have adverse effects on their mental and emotional well-being

1

Age and Gender

We'll analyze the age and gender distribution of marginal workers to identify any patterns or gaps that exist.

2

Industrial Categories Comparison

We'll compare the employment trends of differentindustrial categories to understand their growth and opportunities.

3

Occupational analysis

Explore the types of occupations and industries in which marginal workers are more prevalent. Assess the working conditions and job opportunities in these sectors.

Data collection

Collect relevant data from various sources, such as government surveys, research studies, labor market reports, and organizational data. Ensure that your data is representative and up to date.

Data Cleaning and Preprocessing

- Clean and preprocess the data to handle missing values, outliers, and inconsistencies. This may involve data imputation, standardization, and data transformation as needed.

```
%matplotlib inline
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import datetime
import os
from math import sqrt
import warnings
```

```
## For Multiple Output in single cell
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
warnings.filterwarnings('ignore')
```

```
out_geo = pd.read_csv("C:\\Users\\S.Albert Simion\\
OneDrive\\Desktop\\IBM\\DDW_B06SC_3300_State_TAMIL_NADU-2011.CSV")
out_geo.shape
out_geo.head()
```

```
[9]: (594, 69)
```

[9]:

	Table Code	State Code	District Code	Area Name	Total/ Rural/ Urban	Age group	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	...	Industrial Category - N to O - Females	Industrial Category - P to Q - Persons	Industrial Category - P to Q - Males	Industrial Category - P to Q - Females	Industrial Category - R to U - HHI - Persons	Industrial Category - R to U - HHI - Males	Industrial Category - R to U - HHI - Females
0	B0806SC	`33	`000	State - TAMIL NADU	Total	Total	1200828	589003	611825	221386	...	3565	11080	4019	7061	16833	4266	12567
1	B0806SC	`33	`000	State - TAMIL NADU	Total	`5-14	27791	14125	13666	2447	...	11	122	71	51	427	169	258
2	B0806SC	`33	`000	State - TAMIL NADU	Total	15-34	514340	259560	254780	92423	...	1754	7536	2718	4818	8346	2127	6219
3	B0806SC	`33	`000	State - TAMIL NADU	Total	35-59	542581	251957	290624	99202	...	1619	3205	1131	2074	6591	1487	5104
4	B0806SC	`33	`000	State - TAMIL NADU	Total	60+	115103	62833	52270	27165	...	175	211	93	118	1457	483	974

5 rows × 69 columns

Data Visualization

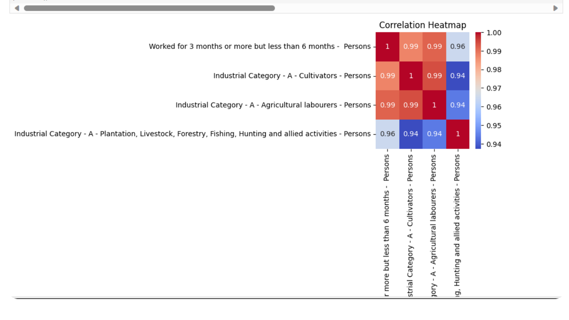
Data Extraction

First, let's extract and load the dataset into a Pandas DataFrame. Make sure to have the Pandas library installed Unleash the power of visual storytelling by incorporating captivating visualizations into your analysis. Learn how to create bar charts, scatter plots, and interactive maps to bring your data to life. Engage your audience and make meaningful connections.

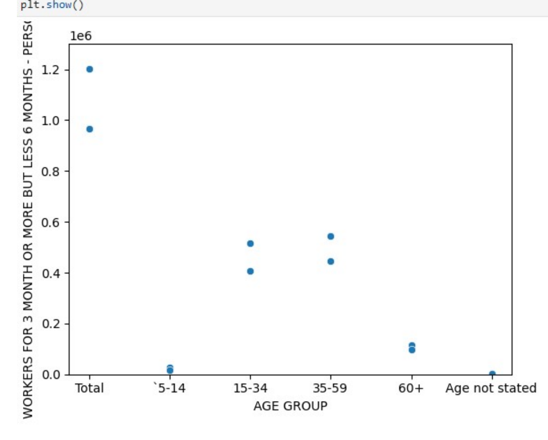
Data Preprocessing

Now, we'll preprocess the data to prepare it for visualization. Preprocessing can include cleaning, filtering, and aggregating the data.Leave a lasting impact with your data-driven conclusions We group the data based on age and working periods, then create a stacked bar chart to visualize the count of workers in different age groups for each working period

```python
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load your dataset
data = pd.read_csv('DDW_B06SC_3300_State_TAMIL_NADU-2011.CSV')

# Create a correlation matrix
correlation_matrix = data[['Worked for 3 months or more but less than 6 months -  Persons','Industrial Category - A - Cultivators - Persons','Industrial

# Create a heatmap
plt.figure(figsize=(3, 3))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```




```python
import pandas as pd
import seaborn as sns
df = pd.read_csv('DDW_B06SC_3300_State_TAMIL_NADU-2011.CSV')
x = df['Age group'].loc[0:10]
y =df['Worked for 3 months or more but less than 6 months -  Persons'].loc[0:10]
sns.scatterplot(x=x, y=y)
plt.xlabel('AGE GROUP')
plt.ylabel('WORKERS FOR 3 MONTH OR MORE BUT LESS 6 MONTHS - PERSON ')
plt.ylim(0, 1300000)

plt.show()
```



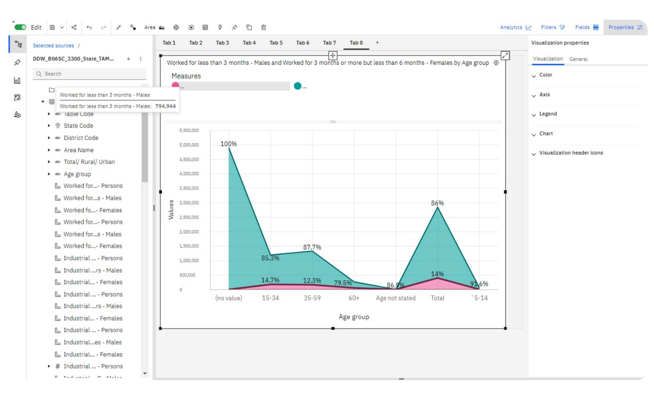
IBM COGNOS IN DATA ANALYTICS

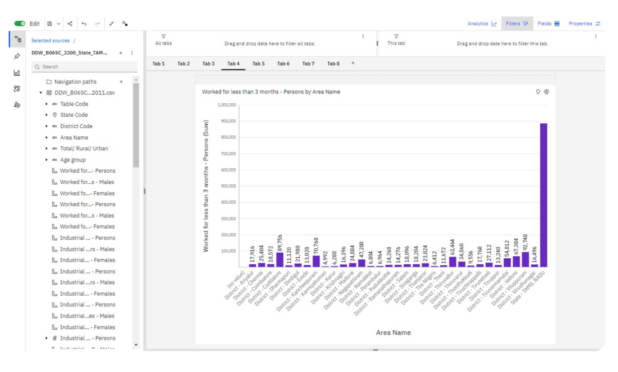
IBM Cognos is a business intelligence and performance management software suite that is used in data analytics and reporting. It provides a range of tools and features for gathering, analyzing, and presenting data to help organizations make informed decisions. Here's how IBM Cognos is used in data analytics

1. Data Integration: IBM Cognos can connect to various data sources, including databases, spreadsheets, cloud-based data, and more. It allows users to consolidate and integrate data from different sources into a unified view for analysis.
2. Data Modeling: Cognos provides tools for data modeling and transforming raw data into a structured format suitable for analysis. Users can create data models and define relationships between data elements.
3. Report Authoring: With Cognos, users can create interactive reports and dashboards. It offers a user-friendly interface for designing reports and customizing the layout, formatting, and visualizations to convey insights effectively.

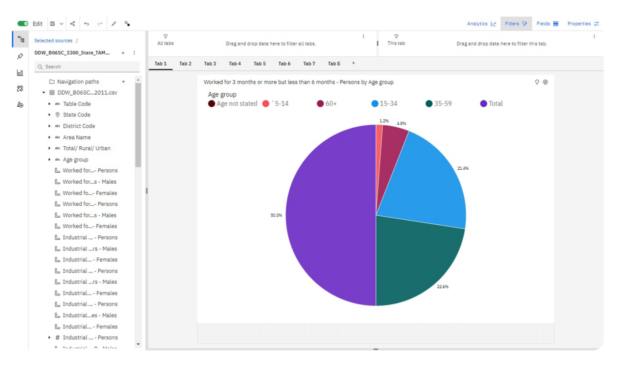
Data Exploration and Analysis: Cognos enables users to explore data through various tools, such as ad-hoc querying and interactive dashboards. Users can drill down into data, apply filters, and perform what-if analysis.

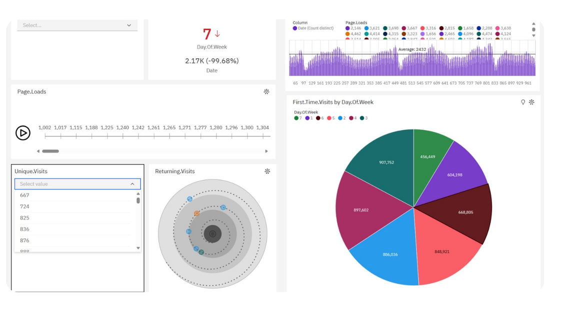
DASHBOARDS (COGNOS)





Dash boards which create datasets into pie chart



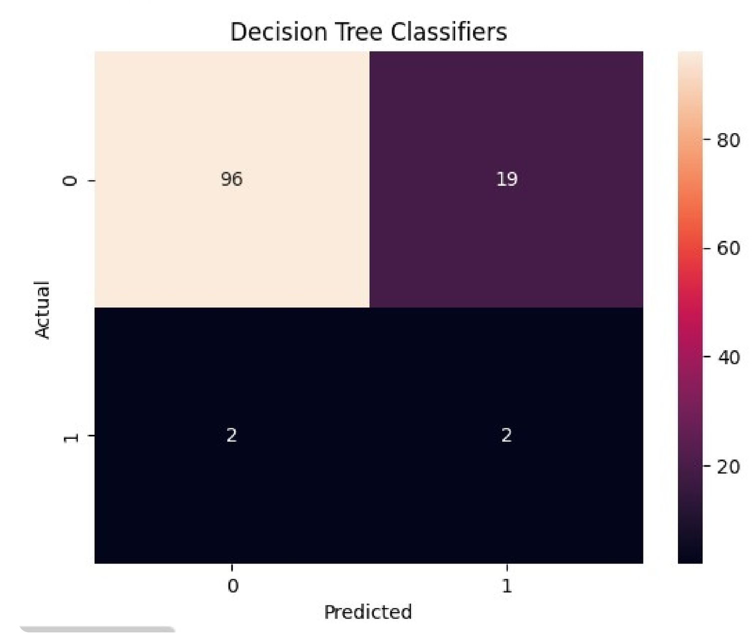


Decision Tree classifiers

```python
# Import necessary libraries
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
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.tree import DecisionTreeClassifier


# Load the CSV dataset
dataset = pd.read_csv('DDW_B06SC_3300_State_TAMIL_NADU-2011.CSV')
# Specify the target column name
target_column = "Industrial Category - G - HHI - Females"  # Replace with the name of your target column

# Identify and encode all categorical columns using one-hot encoding
categorical_columns = dataset.select_dtypes(include=['object']).columns
dataset = pd.get_dummies(dataset, columns=categorical_columns)

# Split the dataset into features and the target variable
X = dataset.drop(columns=[target_column])
y = dataset[target_column]

# Split the dataset into a training set and a test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a Logistic Regression classifier
clf = LogisticRegression()

# Train the classifier on the training data
clf.fit(X_train, y_train)

# Make predictions on the test data
y_pred = clf.predict(X_test)

# Calculate the accuracy of the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```python
# Print classification report
print(classification_report(y_test, y_pred))

# Visualization of the Decision Tree (if the tree is not too deep)
if isinstance(clf, DecisionTreeClassifier) and clf.tree_.max_depth <= 5:
    plt.figure(figsize=(12, 8))
    from sklearn.tree import plot_tree
    plot_tree(clf, filled=True, feature_names=X.columns, class_names=y.unique())
    plt.show()

# Confusion matrix visualization
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Decision Tree Classifiers')
plt.show()

if isinstance(clf, DecisionTreeClassifier):
    feature_importance = clf.feature_importances_
    feature_names = X.columns
    sns.barplot(x=feature_importance, y=feature_names)
    plt.xlabel('Feature Importance')
    plt.ylabel('Feature Name')
    plt.title('Feature Importance')
    plt.show()
```

```
Accuracy: 0.8235294117647058
              precision    recall  f1-score   support

           0       0.98      0.83      0.90       115
           3       0.10      0.50      0.16         4

    accuracy                           0.82       119
   macro avg       0.54      0.67      0.53       119
weighted avg       0.95      0.82      0.88       119
```



There are several popular supervised learning classifiers in Python that you can use for various machine learning tasks. Some of the common ones include:

1. *Logistic Regression*: A simple yet effective binary classification algorithm.
2. *Decision Trees*: Used for both classification and regression problems.
3. *Random Forest*: An ensemble of decision trees, providing more accurate predictions.
4. **Support Vector Machines (SVM)**: Useful for both classification and regression tasks.
5. **K-Nearest Neighbors (K-NN)**: Classifies data points based on the majority class among their k-nearest neighbors.
6. *Naive Bayes*: Particularly suited for text classification tasks.
7. *Neural Networks*: Deep learning models using libraries like TensorFlow or PyTorch.
8. *Gradient Boosting Algorithms*: Such as XGBoost, LightGBM, and CatBoost, are powerful ensemble methods.
9. *AdaBoost*: Another ensemble method that can be used with various weak classifiers.
10. *K-Means*: Often used for clustering, but can be applied to classification by considering the cluster assignments.

To implement these classifiers in Python, you can use libraries like Scikit-Learn, TensorFlow, PyTorch, XGBoost, or LightGBM, depending on the specific classifier and problem you're working on. Here's a high-level example using Scikit-Learn for logistic regression:

python from sklearn.linear_model import LogisticRegression

Create a logistic regression classifier

classifier = LogisticRegression()

Train the classifier on your data

classifier.fit(X_train, y_train)

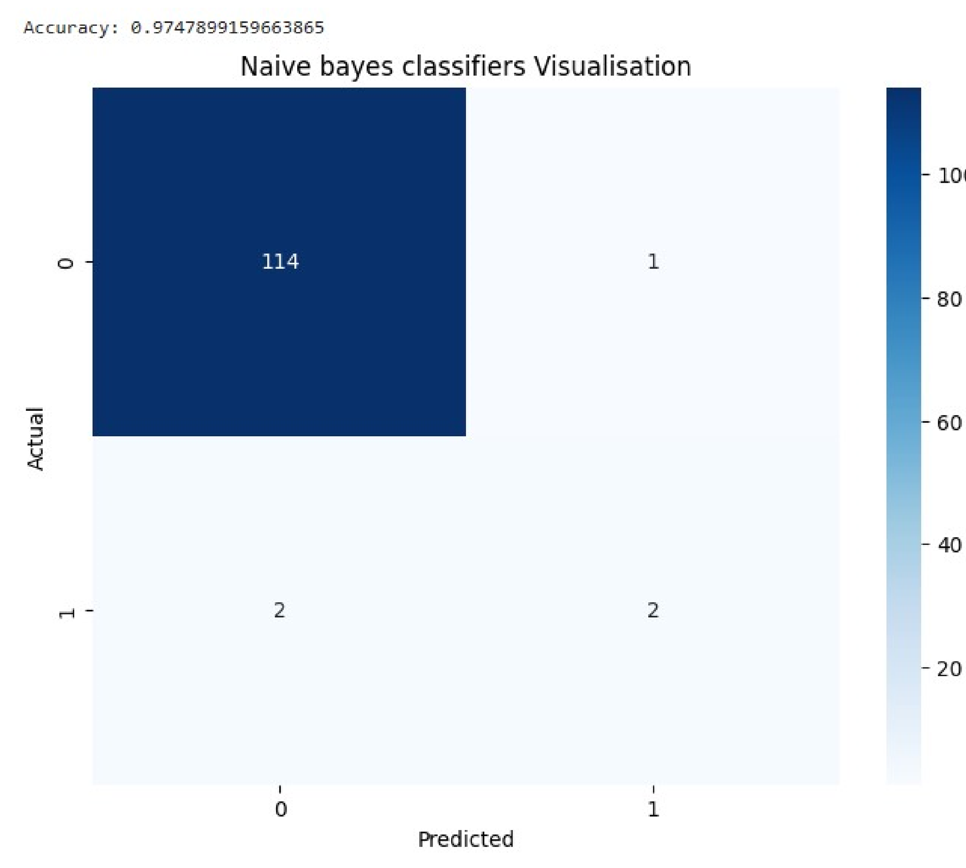
Make predictions

predictions = classifier.predict(X_test)

Make sure to preprocess your data, split it into training and testing sets, and evaluate the performance of your classifier using appropriate metrics like accuracy, precision, recall, or F1-score.

```python
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics

dataset = pd.read_csv('DDW_B06SC_3300_State_TAMIL_NADU-2011.CSV')
# Specify the target column name
target_column = "Industrial Category - G - HHI - Females"  # Replace with the name of your target column

# Identify and encode all categorical columns using one-hot encoding
categorical_columns = dataset.select_dtypes(include=['object']).columns
dataset = pd.get_dummies(dataset, columns=categorical_columns)

# Split the dataset into features and the target variable
X = dataset.drop(columns=[target_column])
y = dataset[target_column]
# Split the dataset into a training and a testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train a Gaussian Naive Bayes classifier
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = nb_classifier.predict(X_test)

# Calculate accuracy
accuracy = metrics.accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Create a confusion matrix
confusion_matrix = metrics.confusion_matrix(y_test, y_pred)

# Visualize the confusion matrix as a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix, annot=True, fmt="d", cmap="Blues")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Naive bayes classifiers Visualisation')
plt.show()
```

```
Accuracy: 0.9747899159663865
```



CONCLUSION

In this analysis, we have examined a dataset pertaining to marginal workers in the state of Tamil Nadu. Our goal was to gain a better understanding of the characteristics of these workers, their distribution across districts, and their working patterns. Here are the main conclusions drawn from our analysis