**IBM PROJECT NAAN MUDHALVAN**

**PHASE 3: DEVELOPMENT**

**TOPIC: PUBLIC HEALTH AWARENESS CAMPAIGN ANALYSIS**

**INTRODUCTION :**

In this phase we done data analysis and machine learning pipeline for a dataset. It encompasses data preprocessing, visualization, and the development of a logistic regression model to make predictions. This part allows to understand how to handle data, explore it visually, and evaluate the performance of a machine learning model using various metrics. It serves as a practical example for data analysis and predictive modeling tasks.

**STEP 1 : IMPORT LIBRARIES**

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt  import seaborn as sns  from sklearn.model\_selection import train\_test\_split  from sklearn.linear\_model import LogisticRegression  from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix  from sklearn.preprocessing import LabelEncoder  from sklearn.impute import SimpleImputer |

In this step, the necessary libraries are imported to work with data, visualize it, and build a machine learning model. These libraries include pandas for data manipulation, matplotlib and seaborn for data visualization, and scikit-learn for machine learning.

**STEP 2 : LOAD DATA**

|  |
| --- |
| f = "survey.csv"  df = pd.read\_csv(f) |

This step loads a dataset from a CSV file named "survey.csv" into a pandas DataFrame called 'df'.

**STEP 3 : DISPLAY BASIC INFORMATION ABOUT THE DATASET**

|  |
| --- |
| df.info() |

This step provides an overview of the dataset's structure, including the number of rows, columns, data types, and information about missing values.

**STEP 4 : HANDLE MISSING VALUES**

|  |
| --- |
| df = df.drop(columns=['comments'])  imputer = SimpleImputer(strategy='median')  df['Age'] = imputer.fit\_transform(df[['Age']]) |

This step first drops the 'comments' column as it's considered not useful for analysis.

Then, it handles missing values in the 'Age' column by filling them with the median value of the 'Age' column.

**STEP 5 : ENCODE BINARY COLUMNS**

|  |
| --- |
| binary\_cols = ['self\_employed', 'family\_history', 'treatment', 'remote\_work', 'tech\_company', 'benefits',  'wellness\_program', 'seek\_help', 'anonymity', 'mental\_health\_interview', 'phys\_health\_interview',  'mental\_vs\_physical', 'obs\_consequence']  for col in binary\_cols:  df[col] = LabelEncoder().fit\_transform(df[col]) |

This step encodes binary columns with '0' and '1' values using LabelEncoder. These columns typically contain 'Yes' or 'No' responses, and they are transformed into numerical values for analysis.

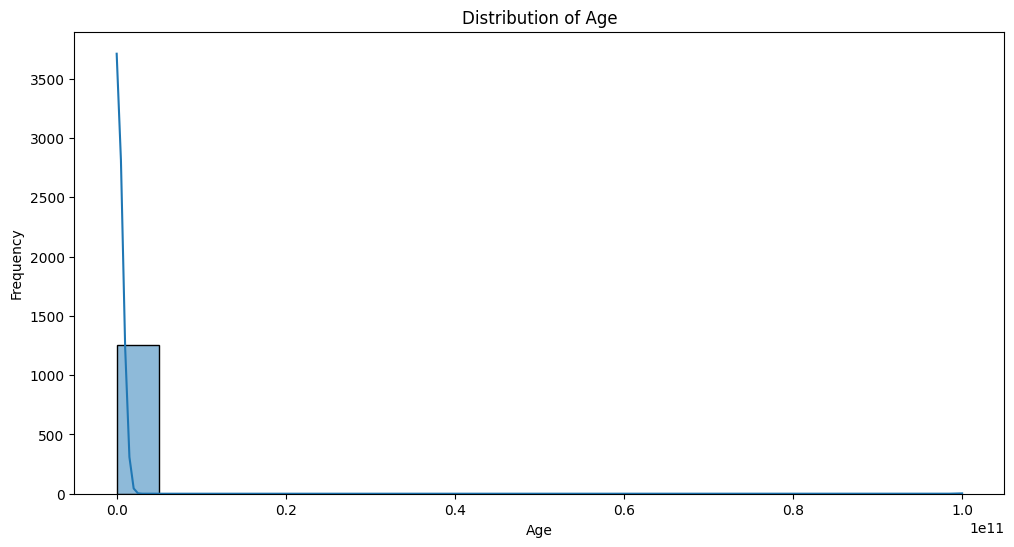
**STEP 6 : DATA VISUALIZATION**

This section includes various data visualization steps using matplotlib and seaborn for understanding the dataset.

**Distribution of Age**

|  |
| --- |
| plt.figure(figsize=(12, 6))  sns.histplot(df['Age'], bins=20, kde=True)  plt.title('Distribution of Age')  plt.xlabel('Age')  plt.ylabel('Frequency')  plt.show() |

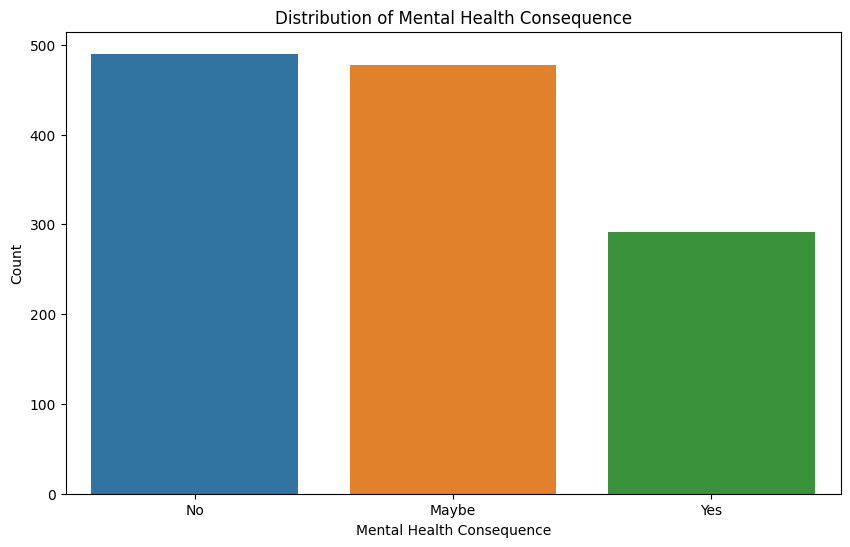
This step creates a histogram and kernel density plot to visualize the distribution of ages in the dataset.



**Distribution of Mental Health Consequences**

|  |
| --- |
| plt.figure(figsize=(10, 6))  sns.countplot(x='mental\_health\_consequence', data=df)  plt.title('Distribution of Mental Health Consequence')  plt.xlabel('Mental Health Consequence')  plt.ylabel('Count')  plt.show() |

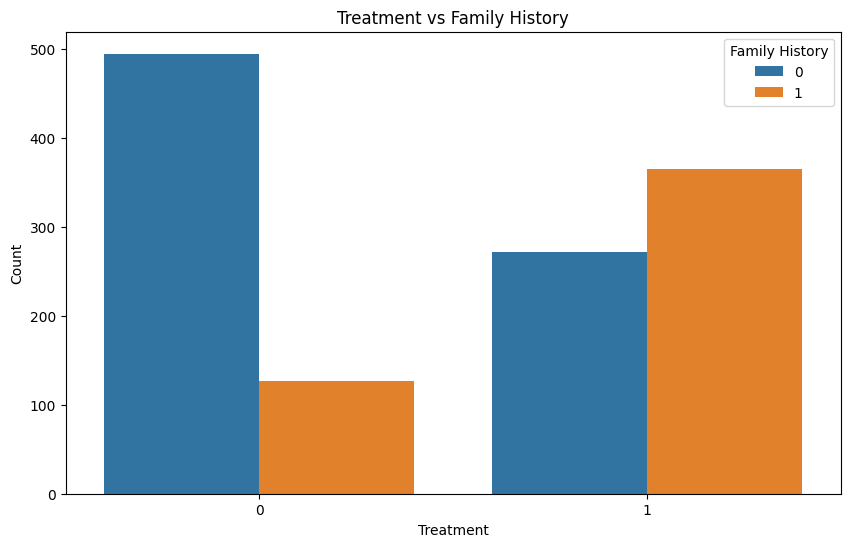
This step creates a count plot to show the distribution of the 'mental\_health\_consequence' column.



**Relationship between Seeking Treatment and Family History**

|  |
| --- |
| plt.figure(figsize=(10, 6))  sns.countplot(x='treatment', hue='family\_history', data=df)  plt.title('Treatment vs Family History')  plt.xlabel('Treatment')  plt.ylabel('Count')  plt.legend(title='Family History', loc='upper right')  plt.show() |

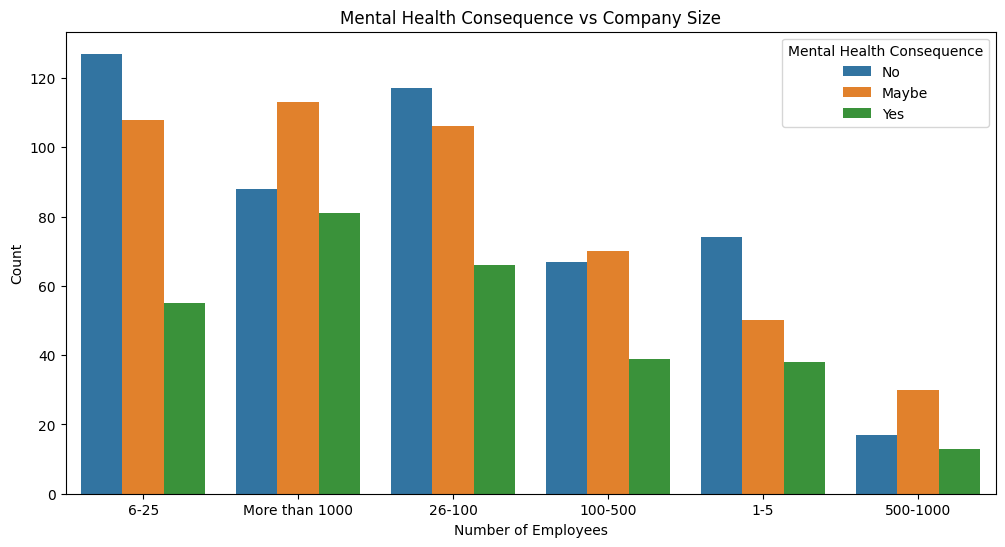
This step creates a count plot to visualize the relationship between seeking treatment and family history.



**Distribution of Mental Health Consequence Based on Company Size**

|  |
| --- |
| plt.figure(figsize=(12, 6))  sns.countplot(x='no\_employees', hue='mental\_health\_consequence', data df)  plt.title('Mental Health Consequence vs Company Size')  plt.xlabel('Number of Employees')  plt.ylabel('Count')  plt.legend(title='Mental Health Consequence', loc='upper right')  plt.show() |

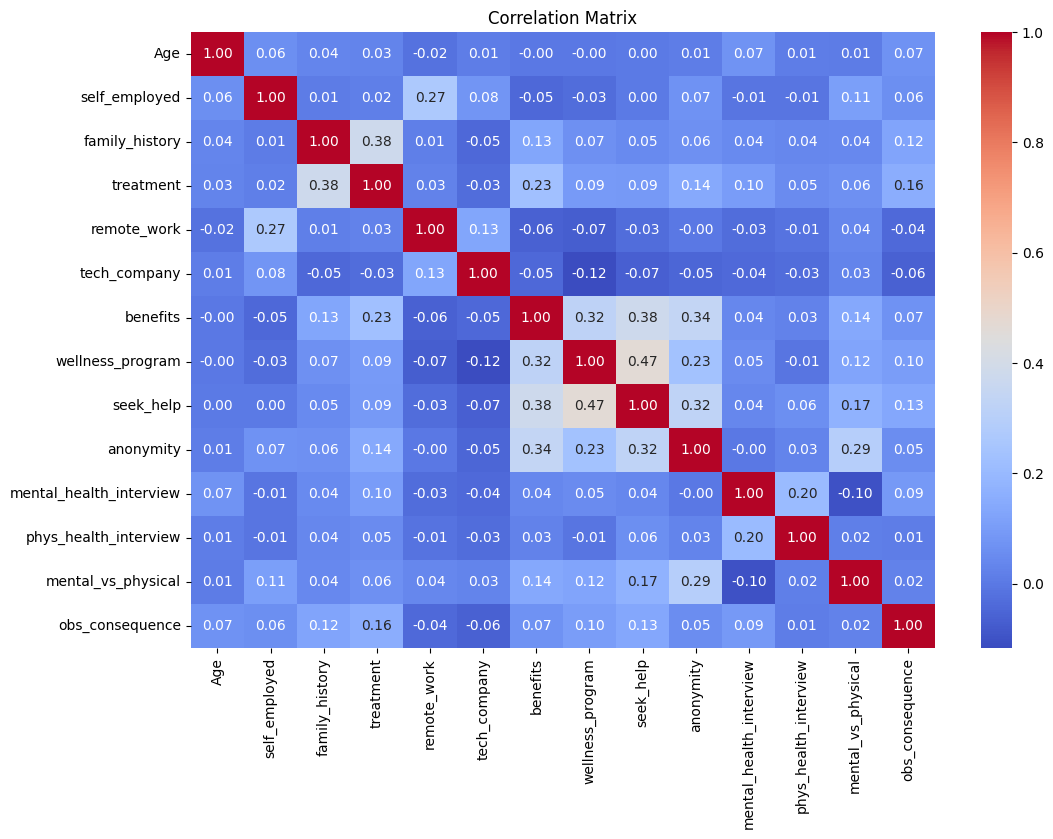
This step creates a count plot to show the distribution of mental health consequences based on company size.



**Correlation Matrix**

|  |
| --- |
| correlation\_matrix = df.corr()  plt.figure(figsize=(12, 8))  sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f")  plt.title('Correlation Matrix')  plt.show() |

This step generates a heatmap to visualize the correlation between numerical variables in the dataset.



**STEP 7 : DATA PREPROCESSING**

|  |
| --- |
| # Drop unnecessary columns  df = df.drop(['Timestamp'], axis=1) # Timestamp is non-numerical  # Split the data into features (X) and target variable (y)  X = df.drop(['treatment'], axis=1)  y = df['treatment']  # Encode categorical variables using one-hot encoding  X = pd.get\_dummies(X, drop\_first=True)  # 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) |

In this step, the code drops the 'Timestamp' column, which is considered non-numerical.It splits the data into features (X) and the target variable (y) and encodes categorical variables using one-hot encoding.

The data is then split into training and testing sets.

**STEP 8 : TRAIN A LOGISTIC REGRESSION MODEL**

|  |
| --- |
| model = LogisticRegression(random\_state=42)  model.fit(X\_train, y\_train) |

This step trains a Logistic Regression model on the training data.

**STEP 9 : MAKE PREDICTIONS AND EVALUATE THE MODEL**

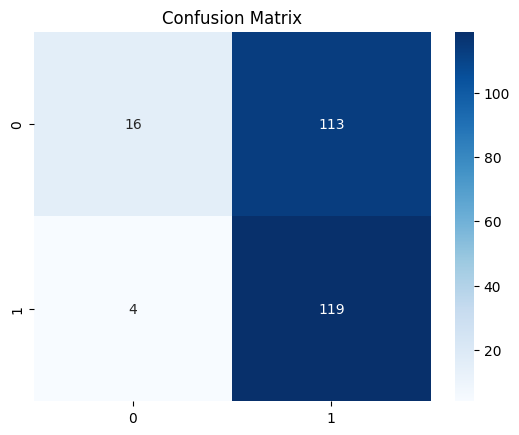
|  |
| --- |
| y\_pred = model.predict(X\_test)  # Evaluate the accuracy  accuracy = accuracy\_score(y\_test, y\_pred)  print(f"Accuracy: {accuracy:.2f}") |

This step makes predictions on the test set and calculates the accuracy of the model.

**STEP 10 : DISPLAY CLASSIFICATION REPORT AND CONFUSION MATRIX**

|  |
| --- |
| print("\nClassification Report:")  print(classification\_report(y\_test, y\_pred))  print("\nConfusion Matrix:")  conf\_matrix = confusion\_matrix(y\_test, y\_pred)  sns.heatmap(conf\_matrix, annot=True, fmt="d", cmap="Blues")  plt.title("Confusion Matrix")  plt.show() |

This step displays a classification report and a confusion matrix to evaluate the performance of the logistic regression model. The classification report provides details on precision, recall, F1-score, and support for each class, while the confusion matrix visually represents the model's performance.



**TEAM MEMBERS:**

SHYAM KUMAR M - 2021115107

SUMATHI K - 2021115112

SIVA GANESH S - 2021115108

SUBASHREE S R - 2021115110

YUKESH S - 2021115331