**Logo

Description automatically generated**

**Olalekan Amodemaja**

**llm22fnc**

M.Sc Business Data Analytics

Bangor Business school (BBS)

**Coding For Business Application**

**For: ABJ-4012 Coding For Business Application**

**2023-24**

**Word count: 1988 words (this doesn’t include codes or references)**

**Coding For Business Application Analysis Report**

Contents

[Introduction 3](#_Toc156355031)

[Data Understanding and Processing: 4](#_Toc156355032)

[Data Overview 4](#_Toc156355033)

[Data Cleaning: 5](#_Toc156355034)

[Understanding the Dataset After Cleaning: 6](#_Toc156355035)

[Data Sampling: 8](#_Toc156355036)

[Data Analysis: 8](#_Toc156355037)

[Impact of Job Background and Educational Level on Customers Response rate; 8](#_Toc156355038)

[Chi-Square test: 8](#_Toc156355039)

[Job and Customer Response Rate: 8](#_Toc156355040)

[Education and Customer Response Rate 8](#_Toc156355041)

[Logistic Regression Model: 9](#_Toc156355042)

[Effect of Job Background and Education levels on Customers' Response Rates: 9](#_Toc156355043)

[Job Background; 9](#_Toc156355044)

[Educational Level: 9](#_Toc156355045)

[Relationship between contact type and customer response 10](#_Toc156355046)

[Contact communication type (contact): 10](#_Toc156355047)

[Relationship between contact communication and customer across all age groups 10](#_Toc156355048)

[Ages between 0-29 10](#_Toc156355049)

[Ages between 30-39 10](#_Toc156355050)

[Ages between 40-49 10](#_Toc156355051)

[Ages between 50-59 10](#_Toc156355052)

[Age group above 60 11](#_Toc156355053)

[Conclusion 11](#_Toc156355054)

[Reference 11](#_Toc156355055)

[Codes 12](#_Toc156355056)

# Introduction

Overview of Dataset: The dataset includes a number of important variables, including age, marital status, employment type, education level, contact communications types, and the client's response to term deposits (subscribed or not). Understanding these aspects can help identify the variables affecting subscriber rates and customer behaviour.

This study provides a thorough examination of the "Bank Marketing" dataset, which includes interactions from a Portuguese banking institution's direct marketing effort. The dataset contains a number of characteristics pertaining to campaign details, client demographics, and term deposit subscription results. With the purpose of improving client targeting and response rates, the bank can use the actionable insights and patterns derived from this analysis to inform its marketing initiatives.

Objective:

The primary focus of the analysis is to use data analytics methods to answer particular questions about how different factors affect consumer response rates. In order to provide useful information for future marketing initiatives, the study also attempts to investigate the connections between subscriber outcomes, communication strategies, and customer attributes.

Approach:

To ensure reliability and accuracy, the analysis entails understanding, cleaning, and preprocessing the dataset. To conduct the ensuing analyses, a random sample of 20% of the data is selected subsequently. We seek to investigate the impact of age groups, education levels, contact communication styles, and employment histories on customer response rates using suitable classification algorithms.

Importance:

Financial organisations need to understand the factors influencing customer response rates in order to create focused campaigns. The analysis will yield insights that will not only improve customer engagement techniques but also optimise resource allocation and increase the efficacy of campaigns.

In order facilitate well-informed decision-making for the banking institution's marketing initiatives, this analysis delves into the data, analyses key features, and derives significant insights.

# Data Understanding and Processing:

Data Overview:

Several characteristics pertaining to customer interactions during marketing campaigns are included in the "Bank Marketing" dataset. Gaining a thorough grasp of the dataset's structure, contents, and any potential discrepancies or missing information is essential before beginning the analysis.

Data Structure:

This structure of data looks to be in the semicolon-separated values (i.e. in CSV format), which is widely used to display tabular data.

Semicolons are used to divide the columns, each of which represents a record. The following explains what each column represents for the given header:

age: Age of each individual

job: Career or position title

marital: Status of marriage

Education: Education level

default: Shows whether or not the individual's credit is in default. housing: Tells you whether the person is in possession of a home loan : Indicates whether or not the individual has a personal loan

contact : Mode of communication

Month : Month of the previous conversation

day\_of\_week: Day of the week they where last contacted

duration: duration of the last contact with the customer measured in seconds

Campaign: The amount of contacts made throughout the campaign's peak days.

PDays: Days that have passed since the client was last contacted from a prior campaign :

poutcome: Result of the prior campaign

emp.var.rate: Rate of fluctuation in employment

Cons.price.idx: Index of consumer price

Cons.conf.idx: Index of consumer confidence

euribor3m: The three-month Euribor rate

nr.employed The number of workers

y: Customers response(i.e Target variable indicating whether or not the customer signed up for a term deposit ("yes" or "no").

# Data Cleaning:

The unique Dataset gotten from the UC Irvine Machine Learning Repository was imported in Python Programming Language using Visual Studio Code. After successfully loading the dataset with the column names specified, we observed that the variables were separated with a Semicolon and there was double quotes in all the values. This raw dataset contains abnormalities therefore, to clean this data, there are few steps taken to clean this data for analysis purpose

* Identify the column names
* Removing double quotes(“) from values represented in the columns and replacing them with an empty string
* Corrected “days mapping” dictionary
* Dropped rows with any missing values in the dataset
* Checked for missing values in the entire data frame
* Ensure each column is represented in the correct format. For example, making sure numerical data is in numerical format.

## Understanding the Dataset After Cleaning:

Various steps were employed to understand the cleaned dataset either through statistical method or visualization methods. The following steps were further taken to to understand the clean data set

* Summary Statistics: This aspect Statistically describes each column that is been represented in the data frame. The result of each column shows the Count, Mean, Standard Deviation, Lowest value, Lower Quartile, Median, Upper Quartile, and Highest Value. This results shows how each column is represented.
* Unique Values: This highlights each column’s unique value. For example, the column name ‘marital’ has unique values of married, single, divorce and unknown
* Histogram: This highlights the frequency distribution of an individual column. Also, we get insights if the variables are normally distributed.
* Analyzed Categorical Variable: In this case, we identified the Categorical columns, looked at the frequency distribution of all the unique values in each column and the run a plot if the frequency distribution.
* Analyzed Numerical Variables: In this case, we evaluated the summary statistics of the numerical values and visualized each numerical data using the histogram, pair plot, violin plot and Correlation Matrix

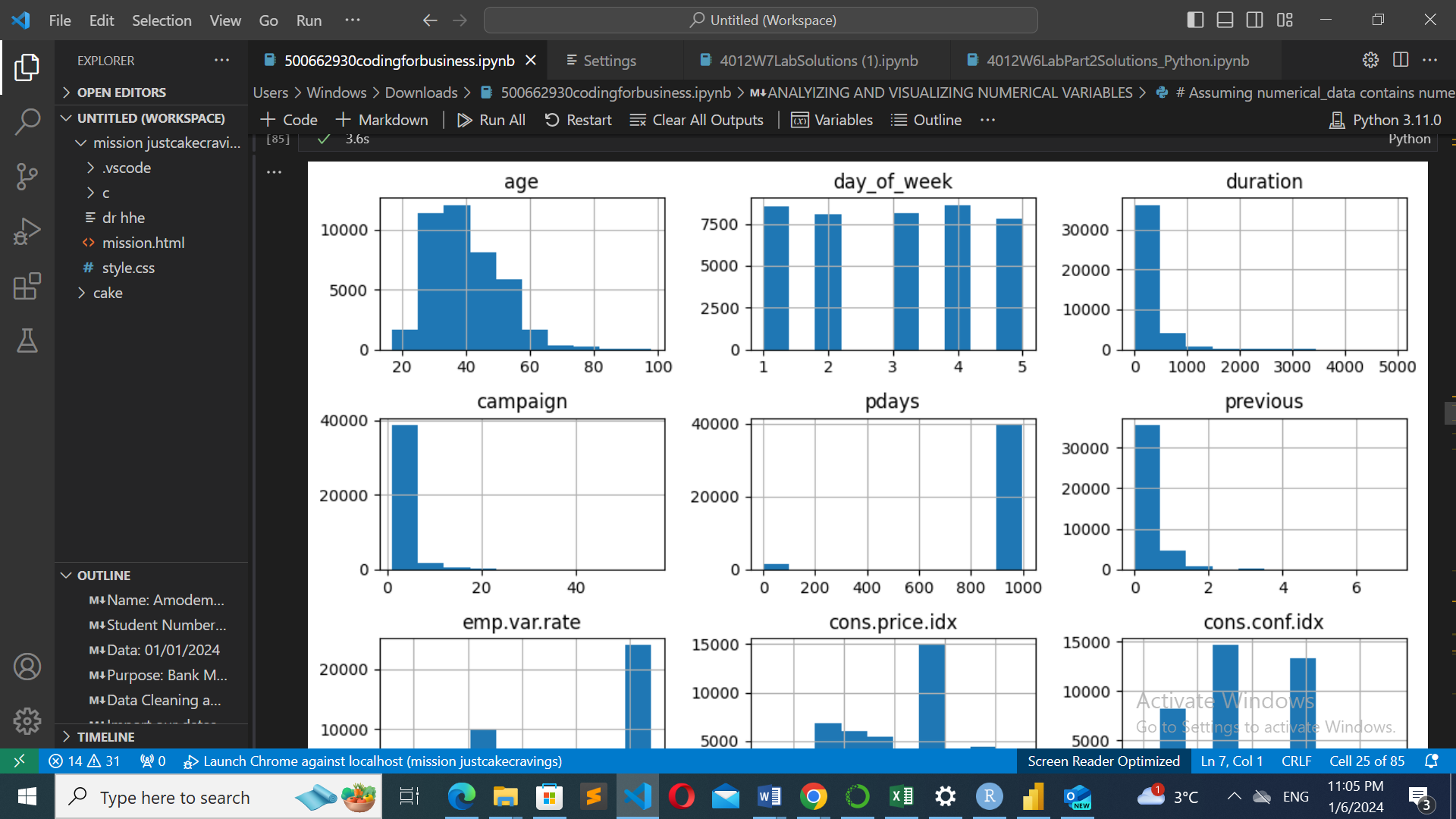


Fig 1.0 Histogram of the numerical data

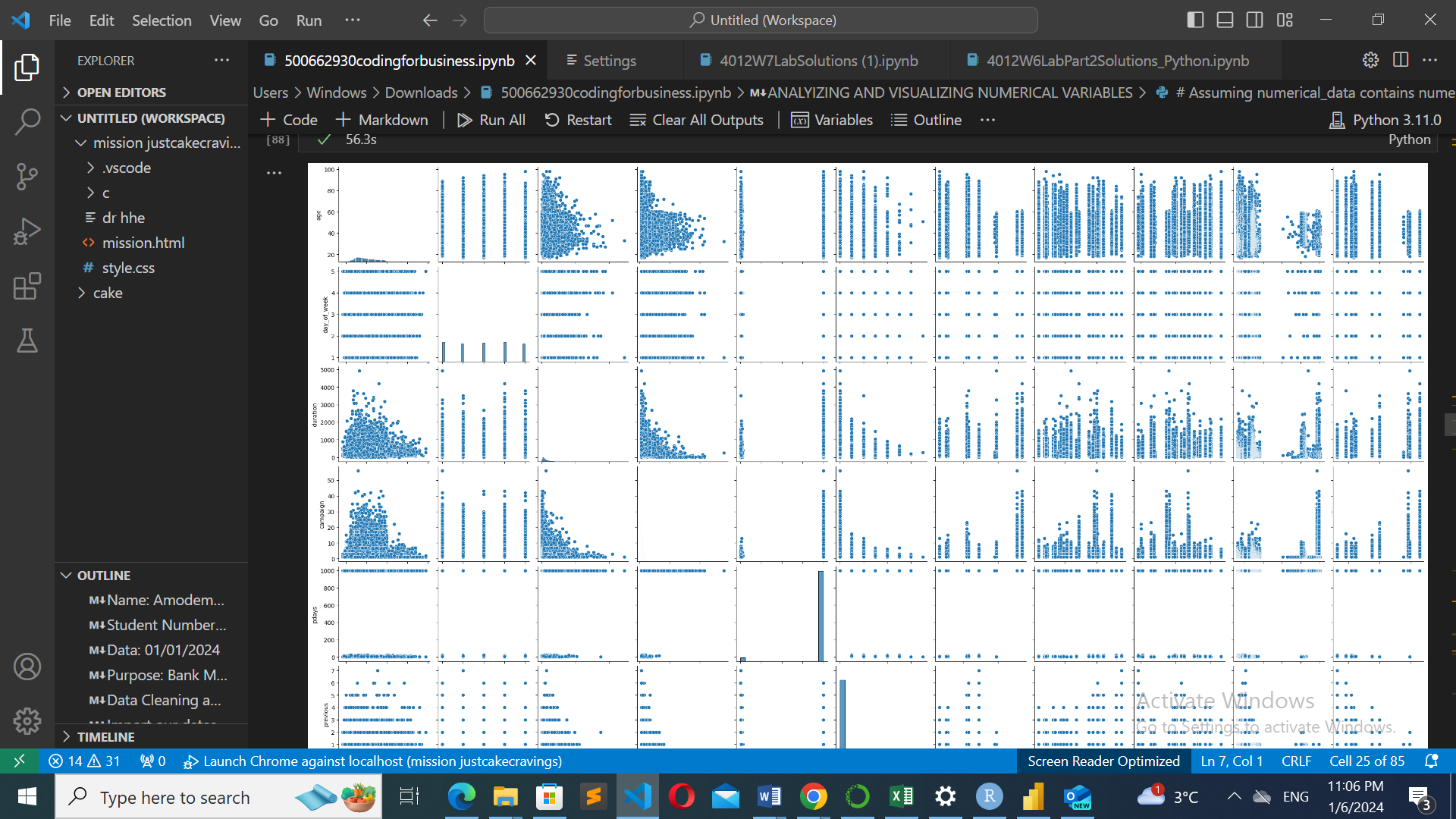


Fig 1.1 Pair plot of the numerical data

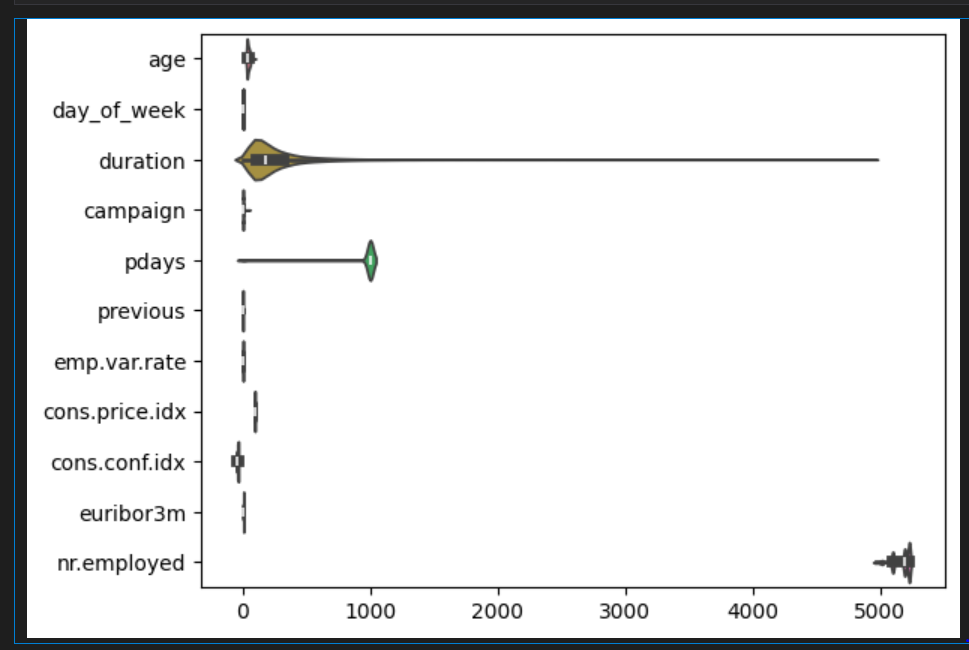


Fig 1.2 Violin plot of the numerical data

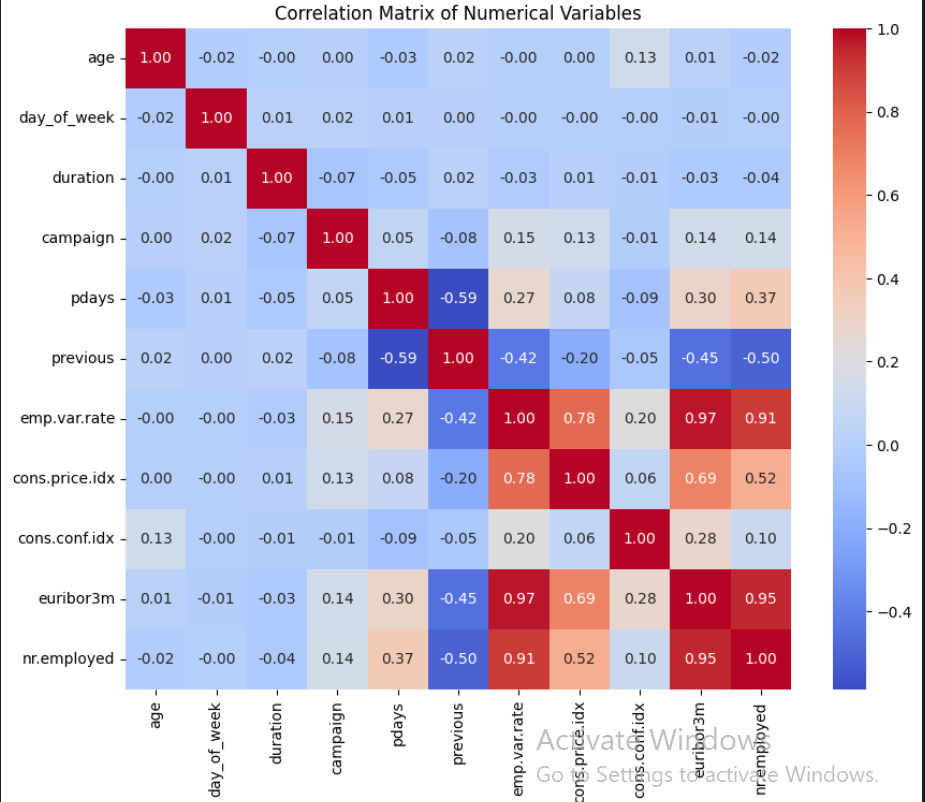


Fig 1.3 Correlation plot of the Numerical Data

# Data Sampling:

This section's goal is to randomly select a 20% subset of the cleaned dataset for additional analysis. A subset of the data is sampled in order to maintain computing efficiency and enable a thorough analysis while maintaining the original dataset's overall representation.

Random sampling technique is the method utilized to extract the subset consisting of 20% of the cleaned data set. Setting a random seed guarantees reproducibility and this gives room to consistent output replication.

# Data Analysis:

After the dataset has been randomly sampled, further steps have been employed to clean the data to make it suitable for analysis. We utilized various data analytics techniques to analyze the data such as the Chi-square test, Logistic regression, and ROC . These Analyses are as follow;

## Impact of Job Background and Educational Level on Customers Response rate;

Chi-Square test: This was performed to know how significant the association between the Job background and educational level affects the customer response rate. Given that the level of significance is 0.05, if the p-value is less than the significance level, this means there is strong evidence against the null hypothesis, which implies that there is an associationbetween the two variables

## Job and Customer Response Rate:

With a relatively large chi-square statistic (218.55) and a p-value (1.03e-40) that is almost zero, we may safely reject the null hypothesis. The findings show that the 'job' and 'y' variables in the dataset have a highly significant correlation.

This statistical significance suggests that there is no independence between the 'job' and 'y' categories. Stated otherwise, an individual's employment status may have significant effects or correlation with the outcome 'y' (which may represent a target variable that, for instance, indicates a person's subscription status for a service).

Education and Customer Response Rate**:**

Considering the chi-square statistic(41,39) is quite high and the p-value(6.81e-07) is very low (almost zero), we may safely rule out the null hypothesis. The findings suggest that the variables 'education' and 'y'(customers’ response rate) in the dataset exhibit a statistically significant correlation.

The statistical significance indicates that there is no independence between the 'education' and 'y'(customers’ response rate) categories. Put in another context, there's a good chance that a person's educational background has a big impact on or correlation with the outcome 'y'(customers’ response rate), which could be whether or not they signed up for a service.

Therefore, based on the result derived from the Chi-square test, it appears that there is a significant association that both Job and Education have with customers' responses (y).

# Logistic Regression Model:

## Effect of Job Background and Education levels on Customers' Response Rates:

### Job Background;

 Interpretation: The result shows the coefficients and odds ratios for various job categories. This illustrates how each job category affects the rate at which customers respond when compared to a reference category, which is typically the baseline category.

For instance: For example, 'job\_retired' has an odds ratio of 2.5919. Its coefficient is 0.9524. This indicates that, after adjusting for other model variables, people classified as "retired" are projected to have roughly 2.59 times higher probability of responding positively compared to the baseline group (such as "job admin").

In summary, some job categories, like "retired," may have a positive impact on consumers' response rates, whilst other job categories may have varied effects, either positive or negative, in relation to the reference category.

### Educational Level:

Interpretation: The coefficients and odds ratios for various education levels, highlights similarities to the job backgrounds. It shows how each education category affects the customers' responding rate in relation to a reference category (such as 'education\_basic.4y' or another selected as the reference). 'Education\_professional.course', for instance, has an odds ratio of 1.5751 and a coefficient of 0.4543. This suggests that compared to the reference category, people who have completed a "professional course" are expected to have 1.58 times higher probabilities of responding positively.

In conclusion, certain educational levels, like "professional course," could significantly increase customers' response rates; nevertheless, other educational levels may have different impacts from those observed in the reference category.

## Relationship between contact type and customer response

### Contact communication type (contact):

Interpretation: The 'contact' variable's coefficient and odds ratio show how different modes of communication impact the customers' response rate.

For instance, 'contact' has an odds ratio of 1.1718 and a coefficient of 0.1585, indicating that a particular kind of contact has roughly 1.17 times better odds of a positive response when compared to the reference contact type.

In conclusion, when considering various forms of contact, the style of communication may have a somewhat positive impact on customers' response rates.

## Relationship between contact communication and customer across all age groups

### Ages between 0-29

This results has a coefficient of 1.0348 and a P- value< 0.001 which denotes that the relation between “contact” and “customers response” for this age group is positive i.e it is significant

### Ages between 30-39

This results has a coefficient of 1.0323 and a P- value< 0.001 which denotes that there is a significantly positive relationship between “contact” and “customer response” for this age group.

### Ages between 40-49

This results has a coefficient of 0.8866 and a P- value< 0.001 similar to the other groups, the p-value is less than the level of significance which means that the relationship between the ‘contact and ‘customers response’ is positively significant.

### Ages between 50-59

This result has a coefficient of 0.8740 and a P- value< 0.001. Similar to other age group’s results interpreted above, the relationship between the ‘contact’ and ‘customer responses’ is positively significant.

### Age group above 60

This result has a coefficient of 1.0631 and a P- value of 0.02 it shows that the ‘contact’of this age group (older generation) also has a positive significant relationship with customers' response rate. But in this case we decide to use a different level of significance i.e 0.05 because of the higher p-value

# Conclusion

The Bank Marketing dataset analysis showed a strong correlation between response rates, method of comunication, and customer demographics. Chi-square tests and logistic regression confirmed that there were significant relationships between job backgrounds and educational attainment and customers' responses. When compared to reference categories, different job categories and level of education levels had different effects on response rates.

Additionally, the communication method showed a significant impact on consumer response rates. The contact approach showed a consistently positive association with customers' response rates across all age groups, highlighting its importance in strategies for marketing. In order to maximise client engagement and campaign efficacy, these findings highlight the significance of tailoring campaign depending on job backgrounds, level of education, and preferred communication mediums.

# Reference

Moro,S., Rita,P and Cortez,P (2012). Bank Marketing. UCI Machine Learning Repository. doi:https://doi.org/10.24432/C5K306.

# Codes

**### Name: Amodemaja Olalekan Quzim**

**### Student Number: 500662930**

**### Module code: ABJ 4012**

**### Module Title: Coding for Business application**

**### Data: 01/01/2024**

**### Purpose: Bank Marketing Data Analysis**

**### TASK 1**

**### Data Cleaning and Manipulation**

**### Import our dataset, Check the structure of the data and Clean the data**

**pip install imblearn**

**pip install statsmodels**

**import pandas as pd**

**import seaborn as sns**

**import random as rand**

**import os**

**column\_names = ["age","job", "marital", "education","default","housing","loan","contact","month","day\_of\_week",**

**"duration","campaign","pdays","previous","poutcome","emp.var.rate","cons.price.idx",**

**"cons.conf.idx", "euribor3m","nr.employed","y"]**

**#Import Data from file location**

**Bank\_market\_data = pd.read\_csv(r"C:/Users/Windows/Desktop/Rexercise/bank-additional-full.csv", sep=';', header=None, names = column\_names, skiprows=1, quoting=3)**

**# after importing the data into Bank\_market\_data DataFrame**

**# Remove double quotes from all columns**

**Bank\_market\_data = Bank\_market\_data.applymap(lambda x: x.replace('"', '') if isinstance(x, str) else x)**

**print(Bank\_market\_data.head())**

**print(Bank\_market\_data.tail())**

**print(Bank\_market\_data.tail())**

**Bank\_market\_data.dtypes**

**# Corrected days\_mapping dictionary**

**days\_mapping = {'mon': 1, 'tue': 2, 'wed': 3, 'thu': 4, 'fri': 5}**

**# Map the abbreviated day names to numerical values and convert the column to int64**

**Bank\_market\_data['day\_of\_week'] = Bank\_market\_data['day\_of\_week'].map(days\_mapping).astype('int64')**

**# Drop rows with any missing values in Bank\_market\_data and store the cleaned DataFrame in Bank\_market\_clean**

**Bank\_market\_data.dropna(axis=0, inplace=True)**

**# Assign the cleaned DataFrame to Bank\_market\_clean i**

**Bank\_market\_clean = Bank\_market\_data.copy()**

**# Print information about the cleaned DataFrame**

**print(Bank\_market\_clean.info())**

**# Check for missing values in the entire DataFrame**

**missing\_values = Bank\_market\_data.isnull()**

**# Check for non-missing values in the entire DataFrame**

**non\_missing\_values = Bank\_market\_data.notnull()**

**# Display information about the DataFrame including non-null counts**

**Bank\_market\_data.info()**

**# Generate summary statistics for numerical columns**

**summary = Bank\_market\_data.describe()**

**#show missing values if any**

**print(missing\_values)**

**#summary stat and insights of the cleaned data**

**print(Bank\_market\_clean.describe())**

**# Using Bank\_market\_clean as our cleaned DataFrame**

**# Loop through each column and print unique values**

**for column in Bank\_market\_clean.columns:**

**unique\_values = Bank\_market\_clean[column].unique()**

**print(f"Unique values in '{column}':\n{unique\_values}\n")**

**### VISUALIZATION**

**import matplotlib.pyplot as plt**

**# Histogram of age column**

**plt.hist(Bank\_market\_clean['age'], bins=20, color='green', edgecolor='black') # Adjust the number of bins for better visualization**

**plt.title('Histogram of Age')**

**plt.xlabel('Age')**

**plt.ylabel('Frequency')**

**plt.grid(axis='y', alpha=0.75)**

**plt.show()**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**# Using 'age' as the column name in Bank\_market\_clean DataFrame**

**sns.histplot(data=Bank\_market\_clean, x='age', bins=20, kde=True, color='lightgreen') # Adjust the number of bins for better visualization**

**plt.title('Histogram of Age')**

**plt.xlabel('Age')**

**plt.ylabel('Frequency')**

**plt.show()**

**### ANALYZING AND VISUALIZING CATEGORICAL VARIABLE**

**# As Bank\_market\_clean is our DataFrame**

**categorical\_data = Bank\_market\_clean.select\_dtypes(include='object') # Selecting columns with object (categorical) data type**

**# Display the names of the identified categorical columns**

**print("Categorical Variables:")**

**print(categorical\_data.columns)**

**# As 'categorical\_data' contains the identified categorical columns**

**for column in categorical\_data.columns:**

**print(f"\nFrequency distribution of '{column}':")**

**print(categorical\_data[column].value\_counts())**

**print("-" \* 30)**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**# Using 'categorical\_data' as the identified categorical columns**

**for column in categorical\_data.columns:**

**plt.figure(figsize=(8, 6))**

**sns.countplot(x=column, data=categorical\_data, hue= column, palette='viridis')**

**plt.title(f'Count of {column}')**

**plt.xlabel(column)**

**plt.ylabel('Count')**

**plt.xticks(rotation=45)**

**plt.show()**

**# Using 'categorical\_data' which contains the identified categorical columns and 'y' is the dependent variable**

**for column in categorical\_data.columns:**

**cross\_tab = pd.crosstab(categorical\_data[column], Bank\_market\_clean['y'])**

**print(f"\nCross-tabulation between '{column}' and 'y':")**

**print(cross\_tab)**

**print("-" \* 30)**

**#plot categorical features**

**for categorical\_feature in categorical\_data:**

**sns.catplot(x='y', col=categorical\_feature, kind='count', hue = 'y', legend = False, data=Bank\_market\_clean, palette='viridis')**

**plt.show()**

**# as 'y' is the dependent variable and 'job' is a categorical predictor variable**

**cross\_tab = pd.crosstab(Bank\_market\_clean['job'], Bank\_market\_clean['y'])**

**print(cross\_tab)**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**# Example with 'job' and 'y' variables**

**sns.countplot(x='job', hue='y', data=Bank\_market\_clean)**

**plt.title('Count of y by job')**

**plt.xlabel('Job')**

**plt.ylabel('Count')**

**plt.xticks(rotation=45)**

**plt.show()**

**pip install scipy**

**from scipy.stats import chi2\_contingency**

**# Example with 'job' and 'y' variables**

**cross\_tab = pd.crosstab(Bank\_market\_clean['job'], Bank\_market\_clean['y'])**

**chi2, p, dof, expected = chi2\_contingency(cross\_tab)**

**print(f"Chi-Square Statistic: {chi2}")**

**print(f"P-value: {p}")**

**### ANALYIZING AND VISUALIZING NUMERICAL VARIABLES**

**# Assuming numerical\_data contains numerical columns**

**numerical\_data = Bank\_market\_clean.select\_dtypes(include='number') # Selecting columns with object (categorical) data type**

**summary\_stats = numerical\_data.describe()**

**print(summary\_stats)**

**# Assuming numerical\_data contains numerical columns**

**numerical\_data\_head = Bank\_market\_clean.select\_dtypes(include='number').head()**

**print(numerical\_data\_head)**

**import matplotlib.pyplot as plt**

**# Assuming 'numerical\_data' contains numerical columns**

**numerical\_data.hist(figsize=(10, 8))**

**plt.tight\_layout()**

**plt.show()**

**import seaborn as sns**

**# Assuming 'numerical\_data' contains numerical columns, plot pair column**

**sns.pairplot(numerical\_data)**

**plt.show()**

**import seaborn as sns**

**# Assuming 'numerical\_data' contains numerical columns**

**sns.violinplot(data=numerical\_data, orient='h')**

**plt.show()**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**# using Bank\_market\_clean as our DataFrame containing numerical columns**

**numerical\_data = Bank\_market\_clean.select\_dtypes(include='number') # Selecting numerical columns**

**# Calculate the correlation matrix**

**correlation\_matrix = numerical\_data.corr()**

**# Create a heatmap to visualize the correlation matrix**

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

**sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt='.2f')**

**plt.title('Correlation Matrix of Numerical Variables')**

**plt.show()**

**### Discrete and Continuous Variables**

**# Assuming 'numerical\_data' contains numerical columns**

**for col in numerical\_data.columns:**

**unique\_count = numerical\_data[col].nunique()**

**if unique\_count < 20: # Example threshold, adjust as needed**

**print(f"{col} is a discrete variable with {unique\_count} unique values.")**

**else:**

**print(f"{col} is a continuous variable.")**

**#Count for Discrete data**

**discrete\_data =[feature for feature in numerical\_data if len(Bank\_market\_clean[feature].unique())<30]**

**print('Discrete Data Count: {}'. format(len(discrete\_data)))**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**plt.figure(figsize=(20,60), facecolor='white')**

**plotnumber =1**

**# Box plot for discrete values**

**for feature in discrete\_data:**

**ax = plt.subplot(12,3,plotnumber)**

**sns.boxplot(x= 'y', y= Bank\_market\_clean[feature], data = Bank\_market\_clean)**

**plt.title('Box Plot of Discrete Variable')**

**plt.xlabel(feature)**

**plotnumber+=1**

**plt.show()**

**continuous\_data =[feature for feature in numerical\_data if len(Bank\_market\_clean[feature].unique())<30]**

**print('Continuous Data Count: {}'. format(len(continuous\_data)))**

**plt.figure(figsize=(20,60), facecolor='white')**

**plotnumber =1**

**# Box plot for discrete values**

**for feature in continuous\_data:**

**ax = plt.subplot(12,3,plotnumber)**

**sns.boxplot(x= 'y', y= Bank\_market\_clean[feature], data = Bank\_market\_clean)**

**plt.title('Box Plot of Continuous Variable')**

**plt.xlabel(feature)**

**plotnumber+=1**

**plt.show()**

**### SAMPLING THE DATASET**

**import pandas as pd**

**# Set a random seed for reproducibility**

**random\_seed = 42**

**# Randomly extract 20% of the data and store it in a new DataFrame**

**sampled\_data = Bank\_market\_clean.sample(frac=0.2, random\_state=random\_seed)**

**# Display the shape or basic information of the sampled data**

**print("Shape of Sampled Data:", sampled\_data.shape)**

**# Now 'sampled\_data' contains 20% of the original 'Bank\_market\_clean' data for subsequent analysis**

**sampled\_data.head()**

**#group sampled data using y column**

**sampled\_data['y'].groupby(sampled\_data['y']).count()**

**sampled\_data.dtypes**

**# print unique value of the sampled dataset**

**for column in sampled\_data.columns:**

**unique\_values = sampled\_data[column].unique()**

**print(f"Unique values in '{column}':")**

**print(unique\_values)**

**print("-" \* 30)**

**#### REMOVING UNKNOWN VALUES AND COVERTING YES AND NO TO 0 and 1**

**# using 'sampled\_data' as our DataFrame and 'column\_name' is the column where 'unknown' values exist**

**# using 'sampled\_data' is your DataFrame**

**sampled\_data\_new = sampled\_data[(sampled\_data['loan'] != 'unknown') & (sampled\_data['default'] != 'unknown') & (sampled\_data['housing'] != 'unknown')]**

**# Check unique values in the 'default' column after dropping 'unknown' values**

**print("Unique values in 'default' column:")**

**print(sampled\_data\_new['default'].unique())**

**# using 'column\_name' as the column where 'unknown' values existed**

**print(sampled\_data\_new['loan'].unique())**

**# or**

**print(sampled\_data\_new['loan'].value\_counts())**

**print(sampled\_data\_new['default'].unique())**

**# or**

**print(sampled\_data\_new['default'].value\_counts())**

**print(sampled\_data\_new['housing'].unique())**

**# or**

**print(sampled\_data\_new['housing'].value\_counts())**

**# 'sampled\_data' is the DataFrame containing 'No' and 'Yes' values in certain columns**

**columns\_to\_convert = ['default','housing', 'loan', 'y'] # Replace 'column1', 'column2' with actual column names**

**# Mapping 'No' to 0 and 'Yes' to 1 in selected columns**

**for col in columns\_to\_convert:**

**sampled\_data\_new[col].replace ({'yes' : 1, 'no': 0}, inplace=True)**

**# Assuming 'sampled\_data\_new' is your DataFrame**

**sampled\_data\_new['contact'] = sampled\_data\_new['contact'].map({'cellular': 0, 'telephone': 1})**

**# Check unique values in the 'contact' column after conversion**

**print("Unique values in 'contact' column:")**

**print(sampled\_data\_new['contact'].unique())**

**# Using 'sampled\_data\_new' as our sampled dataset DataFrame**

**for column in sampled\_data\_new.columns:**

**unique\_values = sampled\_data\_new[column].unique()**

**print(f"Unique values in '{column}':")**

**print(unique\_values)**

**print("-" \* 30)**

**# Corrected code for printing unique values in the entire DataFrame**

**print("Unique values in the entire DataFrame:")**

**print(sampled\_data\_new.apply(lambda x: x.unique()))**

**# List of columns to convert from 'yes' and 'no' to 1 and 0**

**columns\_to\_convert = ['default','housing', 'loan', 'y']**

**# Dictionary mapping 'yes' and 'no' to 1 and 0**

**mapping = {'yes': 1, 'no': 0}**

**# Apply mapping to the specified columns**

**sampled\_data\_new[columns\_to\_convert] = sampled\_data\_new[columns\_to\_convert].replace(mapping)**

**sampled\_data\_new.head()**

**sampled\_data\_new1 = pd.get\_dummies(data = sampled\_data\_new, columns=['marital', 'job', 'education','month', 'poutcome'])**

**sampled\_data\_new1.columns**

**sampled\_data\_new1.sample(5)**

**# convert boolean columns to integer**

**columns\_to\_convert = ['marital\_divorced', 'marital\_married', 'marital\_single', 'marital\_unknown',**

**'job\_admin.', 'job\_blue-collar', 'job\_entrepreneur', 'job\_housemaid',**

**'job\_management', 'job\_retired', 'job\_self-employed', 'job\_services',**

**'job\_student', 'job\_technician', 'job\_unemployed', 'job\_unknown',**

**'education\_basic.4y', 'education\_basic.6y', 'education\_basic.9y',**

**'education\_high.school', 'education\_illiterate', 'education\_professional.course',**

**'education\_university.degree', 'education\_unknown', 'month\_apr', 'month\_aug',**

**'month\_dec', 'month\_jul', 'month\_jun', 'month\_mar', 'month\_may', 'month\_nov',**

**'month\_oct', 'month\_sep', 'poutcome\_failure', 'poutcome\_nonexistent',**

**'poutcome\_success']**

**sampled\_data\_new1[columns\_to\_convert] = sampled\_data\_new1[columns\_to\_convert].astype(int)**

**pip install scikit-learn**

**scale\_columns = ['age', 'duration', 'pdays', 'campaign', 'emp.var.rate', 'cons.price.idx', 'euribor3m', 'nr.employed']**

**from sklearn.preprocessing import MinMaxScaler**

**# scale columns**

**scaler = MinMaxScaler()**

**sampled\_data\_new1[scale\_columns] = scaler.fit\_transform(sampled\_data\_new1[scale\_columns])**

**for col in sampled\_data\_new1:**

**print(f'{col}: {sampled\_data\_new1[col].unique()}')**

**# TASK 2 : Data Analysis**

**sampled\_data\_new1.dtypes**

**## convert target variable to 0 and 1**

**from scipy.stats import chi2\_contingency**

**# Example with 'job' and 'y' variables**

**cross\_tab = pd.crosstab(sampled\_data\_new['job'], sampled\_data\_new['y'])**

**chi2, p, dof, expected = chi2\_contingency(cross\_tab)**

**print(f"Chi-Square Statistic: {chi2}")**

**print(f"P-value: {p}")**

**print(f"Degrees of freedom: {dof}")**

**import numpy as np**

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

**X = sampled\_data\_new1.drop('y', axis='columns')**

**y = sampled\_data\_new1['y'].astype(np.float32) # Assign 'y' directly**

**# Assuming 'X' contains the predictor variables and 'y' is the target variable**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=15, stratify=y)**

**y\_train.value\_counts()**

**y.value\_counts()**

**y\_test.value\_counts()**

**X\_train.shape**

**X\_test.shape**

**X\_train[:7]**

**pip install RandomUnderSampler**

**from imblearn.under\_sampling import RandomUnderSampler**

**# As 'X' contains features and 'y' is the target variable**

**rus = RandomUnderSampler(random\_state=42)**

**sampled\_data\_new1 = X\_resampled, y\_resampled = rus.fit\_resample(X, y)**

**# Assuming 'y' is the target variable**

**print("Before Random Over/Under Sampling:")**

**print(y.value\_counts())**

**# After resampling**

**print("\nAfter Random Over/Under Sampling:")**

**print(pd.Series(y\_resampled).value\_counts())**

**### SPLITTING RESAMPLED DATA INTO TRAIN AND TEST**

**#### To run Logistic regression to check if the Job background and Education levels afftect Customer's Response**

**#### To check if there is a relationship between contact and customers' response**

**#### To check if the communication to response relationship identified is the same for all age groups**

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

**# Assuming 'X\_resampled' contains resampled features and 'y\_resampled' is the resampled target variable**

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

**# Now we have our resampled data split into training (80%) and testing (20%) sets**

**y\_train.value\_counts()**

**from sklearn.linear\_model import LogisticRegression**

**from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix**

**# As 'X' contains other features, including 'age\_group', and 'y' is the target variable**

**# Split the data into training and testing sets**

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

**# Initialize the Logistic Regression model**

**logistic\_model = LogisticRegression()**

**# Fit the model on the training data**

**logistic\_model.fit(X\_train, y\_train)**

**# Make predictions on the test set**

**y\_pred = logistic\_model.predict(X\_test)**

**# Evaluate the model**

**print("Accuracy:", accuracy\_score(y\_test, y\_pred))**

**print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))**

**print("\nConfusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))**

**# Add more evaluation metrics as needed**

**#coefficient and intercept of the logistic model**

**coefficients = logistic\_model.coef\_[0]**

**intercept =logistic\_model.intercept\_[0]**

**#show coefficient**

**for feature, coef in zip(X\_train.columns, coefficients):**

**print(f'{feature}:{coef:.4f}')**

**#Display the intecept**

**print(f'Intercept:{intercept:.4f}')**

**# Get the coefficients (log odds)**

**coefficients = logistic\_model.coef\_[0]**

**# Get the feature names**

**feature\_names = X.columns # Replace 'X' with your feature DataFrame**

**# Calculate odds ratios**

**odds\_ratios = np.exp(coefficients)**

**# Create a DataFrame to display the coefficients and odds ratios**

**odds\_ratios\_df = pd.DataFrame({'Feature': feature\_names, 'Coefficient': coefficients, 'Odds Ratio': odds\_ratios})**

**print(odds\_ratios\_df)**

**from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, roc\_auc\_score, roc\_curve**

**# Assuming you have fitted your logistic regression model and made predictions (y\_pred, y\_test)**

**print("Accuracy:", accuracy\_score(y\_test, y\_pred))**

**print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))**

**print("\nConfusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))**

**# ROC Curve and AUC-ROC**

**auc = roc\_auc\_score(y\_test, logistic\_model.predict\_proba(X\_test)[:, 1])**

**fpr, tpr, thresholds = roc\_curve(y\_test, logistic\_model.predict\_proba(X\_test)[:, 1])**

**# 'fpr' is false positive rates, 'tpr' is true positive rates, 'thresholds' are threshold values**

**print("\nAUC-ROC:", auc)**

**import matplotlib.pyplot as plt**

**# after computing the ROC curve (fpr, tpr) and calculated the AUC-ROC (auc) as described earlier**

**plt.figure(figsize=(8, 6))**

**plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (AUC = %0.2f)' % auc)**

**plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--', label='Random Guess')**

**plt.xlabel('False Positive Rate')**

**plt.ylabel('True Positive Rate')**

**plt.title('Receiver Operating Characteristic (ROC) Curve')**

**plt.legend(loc='lower right')**

**plt.show()**

**## AGE GROUP**

**# Define age groups (you can adjust these boundaries as needed)**

**bins = [0, 30, 40, 50, 60, 100]**

**labels = ['0-29', '30-39', '40-49', '50-59', '60+']**

**sampled\_data\_new['age\_group'] = pd.cut(sampled\_data\_new['age'], bins=bins, labels=labels, right=False)**

**#print the dataframe with the new "age\_group" column**

**print(sampled\_data\_new[['age','age\_group']])**

**import statsmodels.api as sm**

**# Assuming 'sampled\_data\_new' is your DataFrame containing 'y', 'contact', and 'age\_group' columns**

**age\_groups = sampled\_data\_new['age\_group'].unique()**

**for age\_group in age\_groups:**

**# Subset data for the specific age group**

**subset\_data = sampled\_data\_new[sampled\_data\_new['age\_group'] == age\_group].copy()**

**# Prepare the data for logistic regression**

**subset\_data['contact'] = subset\_data['contact'].astype('category').cat.codes # Encode 'contact' as numerical**

**# Add a constant term to the predictor variable**

**X = sm.add\_constant(subset\_data['contact'])**

**# Fit logistic regression model**

**logit\_model = sm.Logit(subset\_data['y'], X)**

**result = logit\_model.fit()**

**# Print the summary of the logistic regression analysis**

**print(f"\nLogistic Regression Results for Age Group: {age\_group}\n")**

**print(result.summary())**

**## INSIGHT DRAWN FROM THE DATA ANALYSED**

**import seaborn as sns**

**import matplotlib.pyplot as plt**

**# Assuming 'Bank\_market\_clean' is your cleaned DataFrame**

**plt.figure(figsize=(8, 6))**

**sns.countplot(x='contact', hue='y', data=sampled\_data\_new)**

**plt.title('Customer Response by Contact Method')**

**plt.xlabel('Contact Method')**

**plt.ylabel('Count')**

**plt.legend(title='Response', labels=['No', 'Yes'])**

**plt.show()**

**# Assuming 'df' is your DataFrame containing the 'age' column**

**# Define age groups (you can adjust these boundaries as needed)**

**bins = [0, 30, 40, 50, 60, 100]**

**labels = ['0-29', '30-39', '40-49', '50-59', '60+']**

**sampled\_data\_new['age'] = pd.cut(sampled\_data\_new['age'], bins=bins, labels=labels, right=False)**

**plt.figure(figsize=(10, 6))**

**sns.countplot(x='age\_group', hue='y', data=sampled\_data\_new, order=['0-29', '30-39', '40-49', '50-59', '60+'])**

**plt.title('Customer Response by Age Group')**

**plt.xlabel('Age')**

**plt.ylabel('Count')**

**plt.legend(title='Response', labels=['No', 'Yes'])**

**plt.show()**