PRIFYSGOL
BANGOR
UNIVERSITY

Student Name	**Gelareh Kabiri**
Module Supervisor	**Heather He**
Module title	**Python coding**
Date of Submission	**07/01/2024**

Contents

Introduction

The Bank Marketing Dataset is a comprehensive resource that sheds light on the complex interaction between marketing and banking. It may be accessed through the UCI Machine Learning Repository at https://archive.ics.uci.edu/dataset/222/bank+marketing. This carefully selected dataset provides a thorough examination of consumer interactions in the banking industry, making it an invaluable resource for scholars, analysts, and business professionals. With a wide range of characteristics, the dataset offers insights into many facets of consumer behaviors and how they react to financial institutions' marketing campaigns. It contains information about job profiles, educational backgrounds, and demographics in addition to specifics regarding the communication channels that are employed in marketing efforts. Furthermore, economic indicators that help to provide a comprehensive picture of the economic environment in which these marketing initiatives take place include employment variation rates, consumer pricing indices, and current interest rates. The Bank Marketing Dataset is primarily intended for use in predictive modelling and analytics, but it also makes it easier to investigate patterns and trends that affect consumers' choices to sign up for term deposits. This data is crucial for developing strategic marketing plans, improving client interactions, and allocating resources as efficiently as possible in the banking industry. The UCI Machine Learning Repository hosts this publicly available dataset, which is a useful tool for data scientists, academic researchers, and business professionals. Its legitimacy and applicability for a range of uses, such as machine learning, statistical analysis, and exploratory data analysis, are highlighted by its availability.

Task 1

Importing Data

I imported the data for Task 1 and conducted the data analysis and manipulation procedures for the "Bank Marketing" dataset. The aim was to clean and analyze the data, then concentrate on taking a random 20% selection and analyzing it in further detail.

```
In [1]:
#Importing liabraries
import pandas as pd
import numpy as nd
```

```
In [2]:
print("Date: 2024-12-26")
print("Student ID: 500681622")
print("Purpose: Python Assignment")
```

```
Date: 2024-12-26
Student ID: 500681622
Purpose: Python Assignment
```

TASK_1

```
In [3]:
#Importing dataset
data= pd.read_csv("D:/University/Semester1/Coding/Python project/bank-additional-full.csv")
print(data)
```

```
      age;"job";"marital";"education";"default";"housing";"loan";"contact";"month";"day_of_week";"duration";"campaign";"pday
s";"previous";"poutcome";"emp.var.rate";"cons.price.idx";"cons.conf.idx";"euribor3m";"nr.employed";"y"
0      56;"housemaid";"married";"basic.4y";"no";"no";...
1      57;"services";"married";"high.school";"unknown...
2      37;"services";"married";"high.school";"no";"ye...
3      40;"admin.";"married";"basic.6y";"no";"no";"no...
4      56;"services";"married";"high.school";"no";"no...
...                                                 ...
41183  73;"retired";"married";"professional.course";"...
41184  46;"blue-collar";"married";"professional.cours...
41185  56;"retired";"married";"university.degree";"no...
41186  44;"technician";"married";"professional.course...
41187  74;"retired";"married";"professional.course";"...

[41188 rows x 1 columns]
```

Data Manipulation

It was noticed that the dataset required more processing after importing it since it contained values that were semicolon separated. It was fixed by changing the read_csv() function to use the right delimiter, which produced a cleaner dataset. Delimiters are used in literature to organize the data set for processing (Miller, 2018). To understand the structure of the dataset and spot any potential problems, including errors or missing information, I also looked over it. To make sure a thorough comprehension of the data is essential.

```
In [4]:  print("data preview:")
         print(data.head())

data preview:
  age;"job";"marital";"education";"default";"housing";"loan";"contact";"month";"day_of_week";"duration";"campaign";"pdays";"pre
vious";"poutcome";"emp.var.rate";"cons.price.idx";"cons.conf.idx";"euribor3m";"nr.employed";"y"
0  56;"housemaid";"married";"basic.4y";"no";"no";...
1  57;"services";"married";"high.school";"unknown...
2  37;"services";"married";"high.school";"no";"ye...
3  40;"admin.";"married";"basic.6y";"no";"no";"no...
4  56;"services";"married";"high.school";"no";"no...
```

```
In [5]:  # Data_Manipulation
         data= pd.read_csv("D:/University/Semester1/Coding/Python project/bank-additional-full.csv", delimiter=";", quotechar='"')
         print(data.head())

   age        job  marital    education  default housing loan    contact  \
0   56  housemaid  married     basic.4y       no      no   no  telephone
1   57   services  married  high.school  unknown      no   no  telephone
2   37   services  married  high.school       no     yes   no  telephone
3   40     admin.  married     basic.6y       no      no   no  telephone
4   56   services  married  high.school       no      no  yes  telephone

  month day_of_week  ...  campaign  pdays  previous     poutcome emp.var.rate  \
0   may         mon  ...         1    999         0  nonexistent          1.1
1   may         mon  ...         1    999         0  nonexistent          1.1
2   may         mon  ...         1    999         0  nonexistent          1.1
3   may         mon  ...         1    999         0  nonexistent          1.1
4   may         mon  ...         1    999         0  nonexistent          1.1

   cons.price.idx  cons.conf.idx  euribor3m  nr.employed   y
0          93.994          -36.4      4.857       5191.0  no
1          93.994          -36.4      4.857       5191.0  no
2          93.994          -36.4      4.857       5191.0  no
3          93.994          -36.4      4.857       5191.0  no
4          93.994          -36.4      4.857       5191.0  no

[5 rows x 21 columns]
```

Data Sampling

A random sample of 20% of the cleaned dataset was taken to make further studies easier. To guarantee that the sampling procedure could be repeated, a random seed was chosen. To extract a random subset, the random_state parameter was set to 42 for reproducibility using the sample() method.

In [6]:
```
## Data_sampling
random_seed = 42
sampled_data = data.sample(frac=0.2, random_state=random_seed)
print("Sampled data preview:")
print(sampled_data.head())
```

```
Sampled data preview:
       age          job  marital    education  default housing loan  \
32884   57   technician  married  high.school       no      no  yes
3169    55      unknown  married      unknown  unknown     yes   no
32206   33  blue-collar  married     basic.9y       no      no   no
9403    36       admin.  married  high.school       no      no   no
14020   27    housemaid  married  high.school       no     yes   no

         contact month day_of_week  ...  campaign  pdays  previous  \
32884   cellular   may         mon  ...         1    999         1
3169   telephone   may         thu  ...         2    999         0
32206   cellular   may         fri  ...         1    999         1
9403   telephone   jun         fri  ...         4    999         0
14020   cellular   jul         fri  ...         2    999         0

          poutcome emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  \
32884      failure         -1.8          92.893          -46.2      1.299
3169   nonexistent          1.1          93.994          -36.4      4.860
32206      failure         -1.8          92.893          -46.2      1.313
9403   nonexistent          1.4          94.465          -41.8      4.967
14020  nonexistent          1.4          93.918          -42.7      4.963

       nr.employed   y
32884       5099.1  no
3169        5191.0  no
32206       5099.1  no
9403        5228.1  no
14020       5228.1  no

[5 rows x 21 columns]
```

Task 2

In Task 2, the "Bank Marketing" dataset was explored and analysed to use the proper data analytics approaches to answer queries. Using a classification system to address important questions was the focus of the investigation.

Job backgrounds and Education Levels Analysis

Job Backgrounds and Education Levels Analysis

In [7]:

```python
# Assuming 'job' and 'education' are relevant features for analysis
X1 = sampled_data[['job', 'education']]
y1 = sampled_data['y']

# Splitting the data into training and testing sets
from sklearn.model_selection import train_test_split

X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.2, random_state=42)

# One-hot encoding for categorical variables
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Define the transformer for one-hot encoding
preprocessor1 = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(), ['job', 'education'])
    ],
    remainder='passthrough'
)

# Assuming Logistic Regression as the chosen classification algorithm
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Create a pipeline with one-hot encoding and logistic regression
model1 = Pipeline(steps=[
    ('preprocessor', preprocessor1),
    ('classifier', LogisticRegression(random_state=42))
])

# Training the model
model1.fit(X1_train, y1_train)

# Predicting responses on the test set
predictions1 = model1.predict(X1_test)

# Evaluating the model
accuracy1 = accuracy_score(y1_test, predictions1)
conf_matrix1 = confusion_matrix(y1_test, predictions1)
classification_report1 = classification_report(y1_test, predictions1)

# Displaying results
print(f"Accuracy: {accuracy1}")
print("\nConfusion Matrix:\n", conf_matrix1)
print("\nClassification Report:\n", classification_report1)
```

```
Accuracy: 0.8743932038834952

Confusion Matrix:
 [[1441    0]
 [ 207    0]]

Classification Report:
              precision    recall  f1-score   support

          no       0.87      1.00      0.93      1441
         yes       0.00      0.00      0.00       207

    accuracy                           0.87      1648
   macro avg       0.44      0.50      0.47      1648
weighted avg       0.76      0.87      0.82      1648
```

A machine learning pipeline for response prediction based on "job" and "education"-related features is built in the code that is provided. These are the logical actions performed: The train_test_split function from scikit-learn is used to divide the data into training and testing sets. Column Transformer is used to incorporate the one-hot encoded categorical variables ('job' and 'education') into the preprocessing pipeline. Scikit-learn's Pipeline is used to design a machine learning pipeline. The Pipeline tool from Scikit-learn is a comprehensive Python utility that simplifies the process of building and running machine learning processes. Müller and Guido (2017) describe this capability as allowing users to easily combine various machine learning models and data pretreatment procedures into a cohesive and effective pipeline. The `Pipeline` class streamlines the code structure and improves repeatability by encapsulating different phases of data processing, including data transformation and model training. This methodology ensures a well-organized and systematic approach to machine learning tasks, while also encouraging cleaner and more understandable code and the consistent implementation of a series of operations on the data. A logistic regression classifier and a preprocessor manage one-hot encoding in this pipeline. Using the pipeline's fit approach, the logistic regression model is trained on the training set. The trained model is used to make predictions on the test set. Metrics from the classification report, accuracy, and confusion matrix are used to assess the model's performance. A machine learning model's accuracy, as an indicator of overall correctness, is determined by the ratio of correctly predicted instances to the total instances. The computation of accuracy can be accomplished using the `accuracy_score` function from the scikit-learn library in Python (Haghighi et al., 2018). Additionally, the performance of a classification method is visually summarized through a confusion matrix, which showcases the count of false positives, false negatives, true positives, and true negatives. The generation of this matrix in Python is facilitated by the `confusion_matrix` function in the scikit-learn library (Haghighi et al., 2018).

Part 1's analysis clarifies the substantial influence that customers' educational backgrounds and work histories have on their response rates. Based on the occupational histories and educational levels of the consumers, the study predicts their responses with a reasonable degree of accuracy (87.44% overall). This is achieved using a logistic regression model. When these elements are examined more closely, subtle patterns in their influence are revealed by the model's performance measures. The model demonstrates competence in anticipating negative answers ('no') in a range of educational and professional backgrounds. Its capacity to forecast affirmative answers ('yes') shows unpredictability, though. The 'yes' class's precision, recall, and F1-score metrics shed light on the model's capacity to correctly identify favourable outcomes depending on educational and occupational backgrounds. The results essentially point to a significant correlation between the response rates of the clients and their educational and professional backgrounds. The significance of job background and education level in affecting consumers' likely to reply positively is highlighted by the model's ability to predict positive responses across several categories. To gain a more thorough grasp of these dynamics, greater investigation and refining may provide deeper insights into the precise effects of educational attainment and employment histories on customers' response rates.

Contact Communication Type Analysis

Contact Communication Type Analysis

In [8]:

```
# Assuming 'contact' is a relevant feature for analysis
X2 = sampled_data[['contact']]
y2 = sampled_data['y']

# Splitting the data into training and testing sets
from sklearn.model_selection import train_test_split

X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size=0.2, random_state=42)

# One-hot encoding for the 'contact' column
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Define the transformer for one-hot encoding
preprocessor2 = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(), ['contact'])
    ],
    remainder='passthrough'
)

# Assuming Logistic Regression as the chosen classification algorithm
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Create a pipeline with one-hot encoding and logistic regression
model2 = Pipeline(steps=[
    ('preprocessor', preprocessor2),
    ('classifier', LogisticRegression(random_state=42))
])

# Training the model
model2.fit(X2_train, y2_train)

# Predicting responses on the test set
predictions2 = model2.predict(X2_test)

# Evaluating the model
accuracy2 = accuracy_score(y2_test, predictions2)
conf_matrix2 = confusion_matrix(y2_test, predictions2)
classification_report2 = classification_report(y2_test, predictions2)

# Displaying results
print(f"Accuracy: {accuracy2}")
print("\nConfusion Matrix:\n", conf_matrix2)
print("\nClassification Report:\n", classification_report2)
```

```
Accuracy: 0.8743932038834952

Confusion Matrix:
 [[1441    0]
 [ 207    0]]

Classification Report:
               precision    recall  f1-score   support

          no       0.87      1.00      0.93      1441
         yes       0.00      0.00      0.00       207

    accuracy                           0.87      1648
   macro avg       0.44      0.50      0.47      1648
weighted avg       0.76      0.87      0.82      1648
```

The presented code involves multiple important phases: To begin with, the target variable 'y' was created, and the 'contact' column was chosen for analysis. The next steps were taken like previous question to build the model. This section's analysis shows a clear correlation between the selected contact communication kinds and consumers' response rates. The total accuracy of the logistic regression model used to evaluate this association is about 87.44%. This precision illustrates how well the model works to forecast consumers' reactions depending on the designated contact communication kinds. But a deeper look at the model's performance indicators reveals a significant drawback. High precision, recall, and F1-score metrics show that the model is quite good at predicting negative responses ('no'); nevertheless, it has difficulty correctly predicting positive responses ('yes'). The model may not be able to identify and predict positive outcomes based on contact communication types, as evidenced by the consistently low precision, recall, and F1-score for the 'yes' class throughout analyses. Although there is a clear correlation between the types of contact communications and consumers' response rates, the model's shortcomings in anticipating favourable replies call for additional research and possible improvement.

Age Group Analysis

Age Groups Investigation

In [9]:
```python
# Assuming 'age' and 'contact' are relevant features for analysis
X3 = sampled_data[['age', 'contact']]
y3 = sampled_data['y']

# Splitting the data into training and testing sets
from sklearn.model_selection import train_test_split

X3_train, X3_test, y3_train, y3_test = train_test_split(X3, y3, test_size=0.2, random_state=42)

# One-hot encoding for the 'contact' column
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Define the transformer for one-hot encoding
preprocessor3 = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(), ['contact'])
    ],
    remainder='passthrough'
)

# Assuming Logistic Regression as the chosen classification algorithm
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Create a pipeline with one-hot encoding and logistic regression
model3 = Pipeline(steps=[
    ('preprocessor', preprocessor3),
    ('classifier', LogisticRegression(random_state=42))
])

# Training the model
model3.fit(X3_train, y3_train)

# Predicting responses on the test set
predictions3 = model3.predict(X3_test)

# Evaluating the model
accuracy3 = accuracy_score(y3_test, predictions3)
conf_matrix3 = confusion_matrix(y3_test, predictions3)
classification_report3 = classification_report(y3_test, predictions3)

# Displaying results
print(f"Accuracy: {accuracy3}")
print("\nConfusion Matrix:\n", conf_matrix3)
print("\nClassification Report:\n", classification_report3)
```

```
Accuracy: 0.8743932038834952

Confusion Matrix:
 [[1441    0]
 [ 207    0]]

Classification Report:
               precision    recall  f1-score   support

          no       0.87      1.00      0.93      1441
         yes       0.00      0.00      0.00       207

    accuracy                           0.87      1648
   macro avg       0.44      0.50      0.47      1648
weighted avg       0.76      0.87      0.82      1648
```

The 'age' and 'contact' features are the focus of the code when analysing client reaction. The following steps were taken just like the two previous sets of codes. An analysis of the Age Group Investigation using the logistic regression model indicates a consistent link between contact communication type and answers among different age groups. Regardless of age demographics, the general accuracy of roughly 87.44% is consistent. The consistent results observed in the confusion matrix and classification report correspond to the same patterns for the precision, recall, and F1-score metrics for both 'yes' and 'no' replies across various age categories. But as we can see from the precision, recall, and F1-score values of 0 for the 'yes' class, the model has trouble correctly predicting affirmative replies ('yes') across all age groups. Even while the contact communication type - replies association is consistent, the model's ability to detect good outcomes consistently lags. Even though the association holds true for all age groups, the model's inability to accurately predict positive responses indicates that more research and development are necessary to improve its accuracy in predicting good results in contact communication types for all age groups.

Relationship between Education and Responses

```
In [10]:
#Relationship between education and responses
education_insight = sampled_data.groupby(['education', 'y']).size().unstack(fill_value=0)
print("\nInsight 1: Relationship between education and responses")
print(education insight)
```

```
Insight 1: Relationship between education and responses
y                      no  yes
education
basic.4y              755   83
basic.6y              401   39
basic.9y             1144   93
high.school          1684  217
illiterate              4    1
professional.course   943  124
university.degree    2100  318
unknown               272   60
```

The given code segment examines the relationship between customer reactions and educational attainment. First, the 'education' and 'y' (customer replies) columns are used to group the dataset. The next action is to tally the instances of every distinct combination. The final data is then reorganised into a more understandable pivot table style, with counts acting as the intersections, 'education' values acting as the index, and 'y' values acting as columns. When handling missing data, the fill_value=0 argument is used to replace the missing values with zeros. The code concludes by printing the acquired insights to the console, providing a concise and organised summary of the relationship between various educational levels and client reactions. The analysis provides interesting new information on how responses to the marketing effort and educational level are related. Upon analysis of the data, we find that response patterns vary throughout various educational classifications. With 318 good responses against 2100 negative, customers with a university degree have the highest positive response rate. Though not as much as those with a university degree, those with professional courses also show a positive reaction trend. On the other hand, customers with a basic education background, such as basic.4y, fundamental 6y, and basic.9y tend to display more negative than positive answers. Despite having a very small sample size, the illiterate category receives a positive response. Still, this category has few data points, thus interpretation should be done with caution. Higher education levels are typically linked to more positive answers, while lower education levels show a more mixed pattern. It suggests that education level influences customers' responses. Given the importance of these developments, more research is necessary to fully comprehend the relationships between campaign outcomes and education.

Impact of Campaign Duration on Responses

```
#Impact of campaign duration on responses
sampled_data['duration_category'] = pd.cut(sampled_data['duration'], bins=[0, 100, 200, 300, 400, 500, float('inf')], labels
duration_insight = sampled_data.groupby(['duration_category', 'y']).size().unstack(fill_value=0)
print("\nInsight 2: Impact of campaign duration on responses")
print(duration_insight)
```

```
Insight 2: Impact of campaign duration on responses
y                    no  yes
duration_category
0-100              1995   11
101-200            2388  128
201-300            1277  155
301-400             696  116
401-500             366   88
501+                581  437
```

The study in this code focuses on figuring out how the length of the campaign affects responses. Creating discrete time categories, classifying the data according to these categories and response outcomes, and tabulating the results are the tasks involved. First, the dataset gains a new column called "duration_category." Next, bins 0-100, 101-200, 201-300, 301-400, 401-500, and 501+ are created based on the 'duration' column. The pd.cut function is used to carry out this categorization. According to Unpingco (2021), the `pd.cut` method in Python is a potent tool in the pandas library that is used to bin numerical data into discrete intervals. This technique makes it easier to analyse distribution patterns and trends within various ranges by allowing users to split and classify a continuous variable into predetermined bins. It provides a versatile and effective technique to alter and modify data for statistical and exploratory purposes. It is especially helpful when working with datasets having numerical values that need to be categorised for different analyses.The dataset is then categorised according to the 'duration_category' and the response target variable, 'y'. These groupings are made using the groupby method. Data can be grouped by particular columns in Python with the help of the pandas library's `groupby` function. It helps with operations like aggregation, transformation, and filtering by enabling the deployment of functions to each group independently. This technique is crucial for quickly examining patterns and trends within dataset subsets, offering insightful information on the distribution and properties of the data (Slatkin, 2019). Next, the size method is used to aggregate the data and find the number of occurrences for each group. The data is reshaped using the unstack approach to display the results in a way that is easier to understand. With this transformation, the aggregated data is pivoted, with 'y' representing the rows and 'duration_category' the columns. Zeros are used to fill in any missing values (fill_value=0). The code then outputs the insights it has collected, paying particular attention to how campaign duration affects answers. It is possible to clearly see how various duration categories link to the associated response results thanks to the tabular style. There are only 11 positive responses out of 1995 cases in the '0-100' duration category, with the bulk of responses being negative. Positive replies rise dramatically to 128 as the period approaches the '101-200' range, while negative responses stay higher at 2388. The positive answers rise to 155 in the '201-300' length category as well, suggesting that longer campaign durations may have a beneficial impact. With 1277 occurrences, negative comments still outweigh positive ones even in this range. With differing degrees of positive and negative responses, the tendency is maintained in the ensuing length categories. The '501+' category is particularly noteworthy as it exhibits a significant rise in positive replies (437) relative to negative responses (581). Longer campaign durations may be associated with more favourable replies, according to the results; however, the frequency of negative responses across all duration categories points to a complex link between campaign duration and customer involvement.

References

archive.ics.uci.edu. (n.d.). *UCI Machine Learning Repository*. [online] Available at: https://archive.ics.uci.edu/dataset/222/bank+marketing.

Galli, S. (2020). *Python feature engineering cookbook: over 70 recipes for creating, engineering, and transforming features to build machine learning models*. Birmingham, UK: Packt Publishing.

Haghighi, S., Jasemi, M., Hessabi, S. and Zolanvari, A. (2018). PyCM: Multiclass confusion matrix library in Python. *Journal of Open-Source Software*, 3(25), p.729. doi: https://doi.org/10.21105/joss.00729.

Miller, C. (2018). *Hands-on data analysis with NumPy and pandas: implement Python packages from data manipulation to processing*. Birmingham: Packt Publishing Ltd.

Müller, A.C. and Guido, S. (2017). *Introduction to machine learning with Python: a guide for data scientists*. Beijing: O'reilly.

Raúl Garreta, Moncecchi, G., Hauck, T. and Hackeling, G. (2017). *Scikit-learn: machine learning simplified.* Birmingham, Uk: Packt Publishing.

Slatkin, B. (2019). *Effective Python*. Addison-Wesley Professional.

Unpingco, J. (2021). *Python Programming for Data Analysis*. Springer Nature.

Appendix

Codes

#Importing liabraries

import pandas **as** pd

import numpy **as** nd

In [2]:

print("Date: 2024-12-26")

print("Student ID: 500681622")

print("Purpose: Python Assignment")

Date: 2024-12-26

Student ID: 500681622

Purpose: Python Assignment

TASK_1

In [3]:

#Importing dataset

data= pd**.**read_csv("D:/University/Semester1/Coding/Python project/bank-additional-full.csv")

print(data)

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"

0 56;"housemaid";"married";"basic.4y";"no";"no";...

1 57;"services";"married";"high.school";"unknown...

2 37;"services";"married";"high.school";"no";"ye...

3 40;"admin.";"married";"basic.6y";"no";"no";"no...

4 56;"services";"married";"high.school";"no";"no...

... ...

41183 73;"retired";"married";"professional.course";"...

41184 46;"blue-collar";"married";"professional.cours...

41185 56;"retired";"married";"university.degree";"no...

41186 44;"technician";"married";"professional.course...

41187 74;"retired";"married";"professional.course";"...

[41188 rows x 1 columns]

In [4]:

print("data preview:")

print(data.head())

data preview:

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"

0 56;"housemaid";"married";"basic.4y";"no";"no";...

1 57;"services";"married";"high.school";"unknown...

2 37;"services";"married";"high.school";"no";"ye...

3 40;"admin.";"married";"basic.6y";"no";"no";"no...

4 56;"services";"married";"high.school";"no";"no...

In [5]:

 # Data_Manipulation

data= pd.read_csv("D:/University/Semester1/Coding/Python project/bank-additional-full.csv", delimiter=";", quotechar='"')

print(data.head())

```
   age        job  marital    education  default housing loan    contact  \
0   56  housemaid  married     basic.4y       no      no   no  telephone
1   57   services  married  high.school  unknown      no   no  telephone
2   37   services  married  high.school       no     yes   no  telephone
```

```
3  40    admin.  married    basic.6y      no      no   no  telephone
4  56   services  married  high.school      no      no  yes  telephone


  month day_of_week  ...  campaign  pdays  previous     poutcome emp.var.rate  \
0   may         mon  ...         1    999         0  nonexistent          1.1
1   may         mon  ...         1    999         0  nonexistent          1.1
2   may         mon  ...         1    999         0  nonexistent          1.1
3   may         mon  ...         1    999         0  nonexistent          1.1
4   may         mon  ...         1    999         0  nonexistent          1.1


   cons.price.idx  cons.conf.idx  euribor3m  nr.employed   y
0          93.994          -36.4      4.857       5191.0  no
1          93.994          -36.4      4.857       5191.0  no
2          93.994          -36.4      4.857       5191.0  no
3          93.994          -36.4      4.857       5191.0  no
4          93.994          -36.4      4.857       5191.0  no

[5 rows x 21 columns]
```

In [6]:

```
## Data_sampling
random_seed = 42
sampled_data = data.sample(frac=0.2, random_state=random_seed)
print("Sampled data preview:")
print(sampled_data.head())
```

```
Sampled data preview:
       age          job  marital    education  default housing loan  \
32884   57   technician  married  high.school       no      no  yes
3169    55      unknown  married      unknown  unknown     yes   no
32206   33  blue-collar  married     basic.9y       no      no   no
9403    36       admin.  married  high.school       no      no   no
14020   27    housemaid  married  high.school       no     yes   no

         contact month day_of_week  ...  campaign  pdays  previous  \
32884   cellular   may         mon  ...         1    999         1
3169   telephone   may         thu  ...         2    999         0
32206   cellular   may         fri  ...         1    999         1
```

```
9403   telephone   jun        fri  ...        4    999        0
14020   cellular   jul        fri  ...        2    999        0

          poutcome emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  \
32884      failure        -1.8         92.893         -46.2      1.299
3169   nonexistent         1.1         93.994         -36.4      4.860
32206      failure        -1.8         92.893         -46.2      1.313
9403   nonexistent         1.4         94.465         -41.8      4.967
14020  nonexistent         1.4         93.918         -42.7      4.963

       nr.employed   y
32884       5099.1  no
3169        5191.0  no
32206       5099.1  no
9403        5228.1  no
14020       5228.1  no

[5 rows x 21 columns]
```

TASK_2

Job Backgrounds and Education Levels Analysis

In [7]:

```
# Assuming 'job' and 'education' are relevant features for analysis
X1 = sampled_data[['job', 'education']]
y1 = sampled_data['y']


# Splitting the data into training and testing sets
from sklearn.model_selection import train_test_split


X1_train, X1_test, y1_train, y1_test = train_test_split(X1, y1, test_size=0.2, random_state=42)


# One-hot encoding for categorical variables
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline


# Define the transformer for one-hot encoding
```

```
preprocessor1 = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(), ['job', 'education'])
    ],
    remainder='passthrough'
)


# Assuming Logistic Regression as the chosen classification algorithm
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report


# Create a pipeline with one-hot encoding and logistic regression
model1 = Pipeline(steps=[
    ('preprocessor', preprocessor1),
    ('classifier', LogisticRegression(random_state=42))
])


# Training the model
```

```python
model1.fit(X1_train, y1_train)


# Predicting responses on the test set
predictions1 = model1.predict(X1_test)


# Evaluating the model
accuracy1 = accuracy_score(y1_test, predictions1)
conf_matrix1 = confusion_matrix(y1_test, predictions1)
classification_report1 = classification_report(y1_test, predictions1)


# Displaying results
print(f"Accuracy: {accuracy1}")
print("\nConfusion Matrix:\n", conf_matrix1)
print("\nClassification Report:\n", classification_report1)
```

Accuracy: 0.8743932038834952

Confusion Matrix:

[[1441 0]

[207 0]]

Classification Report:

	precision	recall	f1-score	support
no	0.87	1.00	0.93	1441
yes	0.00	0.00	0.00	207
accuracy			0.87	1648
macro avg	0.44	0.50	0.47	1648
weighted avg	0.76	0.87	0.82	1648

C:\ANA\lib\site-packages\sklearn\metrics_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.

_warn_prf(average, modifier, msg_start, len(result))

C:\ANA\lib\site-packages\sklearn\metrics_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.

  _warn_prf(average, modifier, msg_start, len(result))

C:\ANA\lib\site-packages\sklearn\metrics_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.

  _warn_prf(average, modifier, msg_start, len(result))

Contact Communication Type Analysis

In [8]:

```
# Assuming 'contact' is a relevant feature for analysis

X2 = sampled_data[['contact']]

y2 = sampled_data['y']



# Splitting the data into training and testing sets

from sklearn.model_selection import train_test_split



X2_train, X2_test, y2_train, y2_test = train_test_split(X2, y2, test_size=0.2, random_state=42)



# One-hot encoding for the 'contact' column
```

```
from sklearn.preprocessing import OneHotEncoder

from sklearn.compose import ColumnTransformer

from sklearn.pipeline import Pipeline



# Define the transformer for one-hot encoding

preprocessor2 = ColumnTransformer(

    transformers=[

        ('cat', OneHotEncoder(), ['contact'])

    ],

    remainder='passthrough'

)



# Assuming Logistic Regression as the chosen classification algorithm

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report



# Create a pipeline with one-hot encoding and logistic regression

model2 = Pipeline(steps=[
```

```
    ('preprocessor', preprocessor2),
    ('classifier', LogisticRegression(random_state=42))
])

# Training the model
model2.fit(X2_train, y2_train)

# Predicting responses on the test set
predictions2 = model2.predict(X2_test)

# Evaluating the model
accuracy2 = accuracy_score(y2_test, predictions2)
conf_matrix2 = confusion_matrix(y2_test, predictions2)
classification_report2 = classification_report(y2_test, predictions2)

# Displaying results
print(f"Accuracy: {accuracy2}")
print("\nConfusion Matrix:\n", conf_matrix2)
```

```
print("\nClassification Report:\n", classification_report2)

Accuracy: 0.8743932038834952


Confusion Matrix:

 [[1441    0]

 [ 207    0]]


Classification Report:

              precision    recall  f1-score   support


          no       0.87      1.00      0.93      1441

         yes       0.00      0.00      0.00       207


    accuracy                           0.87      1648

   macro avg       0.44      0.50      0.47      1648

weighted avg       0.76      0.87      0.82      1648
```

C:\ANA\lib\site-packages\sklearn\metrics_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.

  _warn_prf(average, modifier, msg_start, len(result))

C:\ANA\lib\site-packages\sklearn\metrics_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.

  _warn_prf(average, modifier, msg_start, len(result))

C:\ANA\lib\site-packages\sklearn\metrics_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.

  _warn_prf(average, modifier, msg_start, len(result))

Age Groups Investigation

In [9]:

```
# Assuming 'age' and 'contact' are relevant features for analysis

X3 = sampled_data[['age', 'contact']]

y3 = sampled_data['y']



# Splitting the data into training and testing sets

from sklearn.model_selection import train_test_split
```

```python
X3_train, X3_test, y3_train, y3_test = train_test_split(X3, y3, test_size=0.2, random_state=42)


# One-hot encoding for the 'contact' column
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline


# Define the transformer for one-hot encoding
preprocessor3 = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(), ['contact'])
    ],
    remainder='passthrough'
)


# Assuming Logistic Regression as the chosen classification algorithm
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
# Create a pipeline with one-hot encoding and logistic regression

model3 = Pipeline(steps=[

    ('preprocessor', preprocessor3),

    ('classifier', LogisticRegression(random_state=42))

])


# Training the model

model3.fit(X3_train, y3_train)


# Predicting responses on the test set

predictions3 = model3.predict(X3_test)


# Evaluating the model

accuracy3 = accuracy_score(y3_test, predictions3)

conf_matrix3 = confusion_matrix(y3_test, predictions3)

classification_report3 = classification_report(y3_test, predictions3)
```

```
# Displaying results

print(f"Accuracy: {accuracy3}")

print("\nConfusion Matrix:\n", conf_matrix3)

print("\nClassification Report:\n", classification_report3)

C:\ANA\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.

  _warn_prf(average, modifier, msg_start, len(result))

C:\ANA\lib\site-packages\sklearn\metrics\_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.

  _warn_prf(average, modifier, msg_start, len(result))

Accuracy: 0.8743932038834952



Confusion Matrix:

 [[1441    0]

 [ 207    0]]



Classification Report:

              precision    recall  f1-score   support
```

```
         no       0.87      1.00      0.93      1441

        yes       0.00      0.00      0.00       207


   accuracy                           0.87      1648

  macro avg       0.44      0.50      0.47      1648

weighted avg       0.76      0.87      0.82      1648
```

C:\ANA\lib\site-packages\sklearn\metrics_classification.py:1248: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.

  _warn_prf(average, modifier, msg_start, len(result))

In [10]:

```
#Relationship between education and responses

education_insight = sampled_data.groupby(['education', 'y']).size().unstack(fill_value=0)

print("\nInsight 1: Relationship between education and responses")

print(education_insight)


#Impact of campaign duration on responses
```

```
sampled_data['duration_category'] = pd.cut(sampled_data['duration'], bins=[0, 100, 200, 300, 400, 500, float('inf')], labels=['0-100', '101-200', '201-300', '301-400', '401-500', '501+'])

duration_insight = sampled_data.groupby(['duration_category', 'y']).size().unstack(fill_value=0)

print("\nInsight 2: Impact of campaign duration on responses")

print(duration_insight)
```

```
Insight 1: Relationship between education and responses

y                    no  yes

education

basic.4y            755   83

basic.6y            401   39

basic.9y           1144   93

high.school        1684  217

illiterate            4    1

professional.course  943  124

university.degree   2100  318

unknown             272   60


Insight 2: Impact of campaign duration on responses

y                  no  yes
```

duration_category

0-100	1995	11
101-200	2388	128
201-300	1277	155
301-400	696	116
401-500	366	88
501+	581	437