CAPSTONE PROJECT PRESENTATION

ALY6140: ANALYTICS SYSTEMS TECHNOLOGY

Group C:
Ebiere Adegbesan
Abiodun Wale-Adebowale
Lishan Wang
Racheal Chidera Ezeja

TABLE OF CONTENTS

GOALS AND QUESTIONS

We aim to answer the following questions:

- **What is the age distribution of customers in the dataset?**
- **Which job types show the highest success rates in the marketing campaign?**
- **How do economic indicators impact the outcomes of the marketing campaign?**
- **Is there a significant variation in campaign success across different education levels?**
- **Does the presence of a housing or personal loan affect a customer's decision to subscribe to a term deposit?**

DATASET DESCRIPTION

The **bank marketing campaigns** dataset describes the results of marketing campaigns conducted by a bank in Portugal. The campaigns primarily involved direct phone calls, where clients were offered the opportunity to place a term deposit. The outcome of these efforts is captured in the target variable: if the client agrees to place a deposit, the target is marked as 'yes', otherwise as 'no'.

- **Shape:** 41118 rows and 21 columns
- **Lable Y/N:** has the client subscribed to a term deposit? (Binary: 'yes', 'no').
- **Numerical features(9):** age, duration, campaign, pdays, previous, emp.Var.Rate, cons.Price.Idx, cons.Conf.Idx, euribor3m, nr.Employed
- **Categorital features(10):** job, marital, education, default, housing, loan, contact, month, day_of_week, poutcome
- **Missing values:** none

ANALYSIS, DESCRIPTION AND DATA EXTRACTION

```
# Load the dataset
import requests
url = 'https://www.dropbox.com/scl/fi/f2dooq1g1cbojkfpi93dc/Bank-marketing-campaign-CSV.csv?rlkey=dazyqp68mohqlqmo4o8yyrdd8&st=b6
res = requests.get(url)
with open('bank_campaign_sheet.csv','wb') as file:
    file.write(res.content)
```

```
# Read the CSV file into a DataFrame
bank_campaign = pd.read_csv('bank_campaign_sheet.csv')

# Display the first few rows of the DataFrame
bank_campaign.head()
```

HEAD DISPLAY OF THE DATASET

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	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	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no

5 rows × 21 columns

- **DATA SHAPE**: THE NUMBER OF ROWS AND COLUMNS IN THE DATASET.

- **DATATYPES**: VERIFICATION OF THE DATA TYPES FOR EACH FEATURE.

```
Out[6]: age                 int64
        job                object
        marital            object
        education          object
        default            object
        housing            object
        loan               object
        contact            object
        month              object
        day_of_week        object
        duration            int64
        campaign            int64
        pdays               int64
        previous            int64
        poutcome           object
        emp.var.rate      float64
        cons.price.idx    float64
        cons.conf.idx     float64
        euribor3m         float64
        nr.employed       float64
        y                  object
        dtype: object
```

```
In [12]:  bank_campaign.shape

Out[12]: (41188, 21)
```

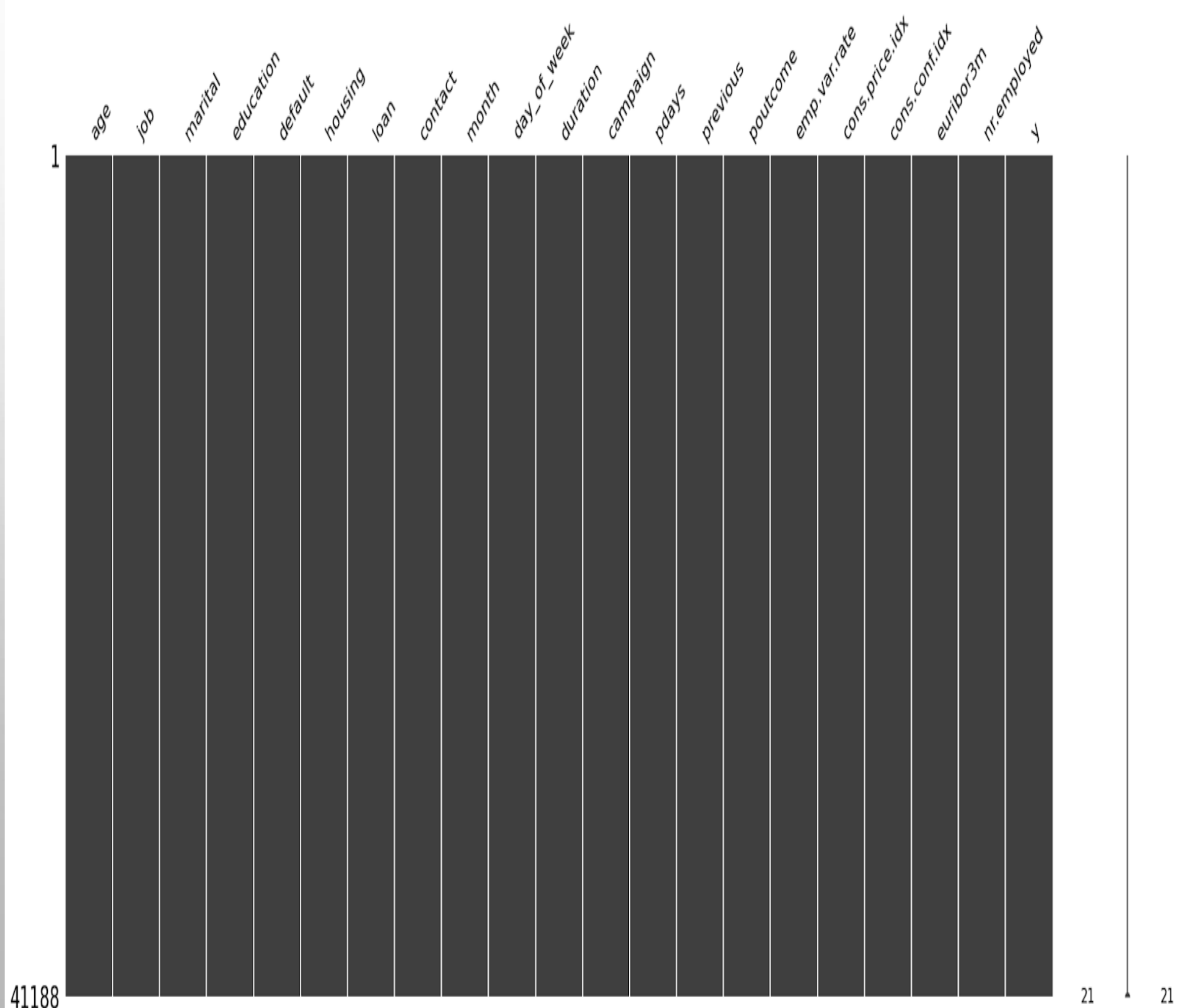
In [9]:

```
# Display summary statistics
bank_campaign.describe()
```

Out[9]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000
mean	40.02406	258.285010	2.567593	962.475454	0.172963	0.081886	93.575664	-40.502600	3.621291	5167.035911
std	10.42125	259.279249	2.770014	186.910907	0.494901	1.570960	0.578840	4.628198	1.734447	72.251528
min	17.00000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.800000	0.634000	4963.600000
25%	32.00000	102.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.700000	1.344000	5099.100000
50%	38.00000	180.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.800000	4.857000	5191.000000
75%	47.00000	319.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.400000	4.961000	5228.100000
max	98.00000	4918.000000	56.000000	999.000000	7.000000	1.400000	94.767000	-26.900000	5.045000	5228.100000

DATA CLEANUP



- **HANDLING MISSING VALUES**: THE DATASET WAS CHECKED FOR MISSING VALUES, BUT NONE WAS PRESENT.

In [10]:

```python
# Check for missing values
print('Data columns with null values:', bank_campaign.isnull().sum(), sep='\n')
```

```
Data columns with null values:
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
```

Outlier detection and handling: Outliers in numerical features were identified using the interquartile range (IQR) method.

```python
# Handling outliers using the IQR method
numerical_features = ['age', 'campaign', 'pdays', 'previous', 'emp.var.rate',
                      'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed']

for feature in numerical_features:
    Q1 = bank_campaign_encoded[feature].quantile(0.25)
    Q3 = bank_campaign_encoded[feature].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    bank_campaign_encoded = bank_campaign_encoded[(bank_campaign_encoded[feature] >= lower_bound) &
                                                  (bank_campaign_encoded[feature] <= upper_bound)]

# Display the updated shape after handling outliers
print("Updated Dataset Shape:", bank_campaign_encoded.shape)
```

```
Updated Dataset Shape: (24919, 55)
```

CATEGORY COUNT

DISTRIBUTION OF CATEGORIES WITHIN EACH CATEGORICAL FEATURE.

```
Job:
job
admin.              10422
blue-collar          9254
technician           6743
services             3969
management           2924
retired              1720
entrepreneur         1456
self-employed        1421
housemaid            1060
unemployed           1014
student               875
unknown               330
Name: count, dtype: int64
----------------------------------------
Marital:
marital
married     24928
single      11568
divorced     4612
unknown        80
Name: count, dtype: int64
----------------------------------------
Education:
education
university.degree       12168
high.school              9515
basic.9y                 6045
professional.course      5243
basic.4y                 4176
basic.6y                 2292
unknown                  1731
illiterate                 18
Name: count, dtype: int64
----------------------------------------
```

```
Default:
default
no          32588
unknown      8597
yes             3
Name: count, dtype: int64
----------------------------------------
Housing:
housing
yes         21576
no          18622
unknown       990
Name: count, dtype: int64
----------------------------------------
Loan:
loan
no          33950
yes          6248
unknown       990
Name: count, dtype: int64
----------------------------------------
Contact:
contact
cellular     26144
telephone    15044
Name: count, dtype: int64
----------------------------------------
Month:
month
may    13769
jul     7174
aug     6178
jun     5318
nov     4101
apr     2632
oct      718
sep      570
mar      546
dec      182
Name: count, dtype: int64
```

```
----------------------------------------
Day_of_week:
day_of_week
thu    8623
mon    8514
wed    8134
tue    8090
fri    7827
Name: count, dtype: int64
----------------------------------------
Poutcome:
poutcome
nonexistent    35563
failure         4252
success         1373
Name: count, dtype: int64
----------------------------------------
Y:
y
no     36548
yes     4640
Name: count, dtype: int64
----------------------------------------
```

BALANCING THE DATASET USING SMOTE

A significant majority (88.73%) of the customers did not subscribe to the term deposit, while only a small fraction (11.27%) did.

After applying SMOTE, both classes (`0` and `1`) have roughly equal representation in the dataset.

```
# Check the distribution of the target variable (showing if the dataset is balanced)
class_distribution = bank_campaign['y_encoded'].value_counts(normalize=True) * 100
print("Class distribution (in percentage):\n", class_distribution)
```

```
Class distribution (in percentage):
 y_encoded
0    88.734583
1    11.265417
Name: proportion, dtype: float64
```

```
from imblearn.over_sampling import SMOTE

# Define features and target
X = bank_campaign.drop(['y', 'y_encoded'], axis=1)
y = bank_campaign['y_encoded']

# One-Hot Encode categorical variables
X = pd.get_dummies(X, drop_first=True)
```

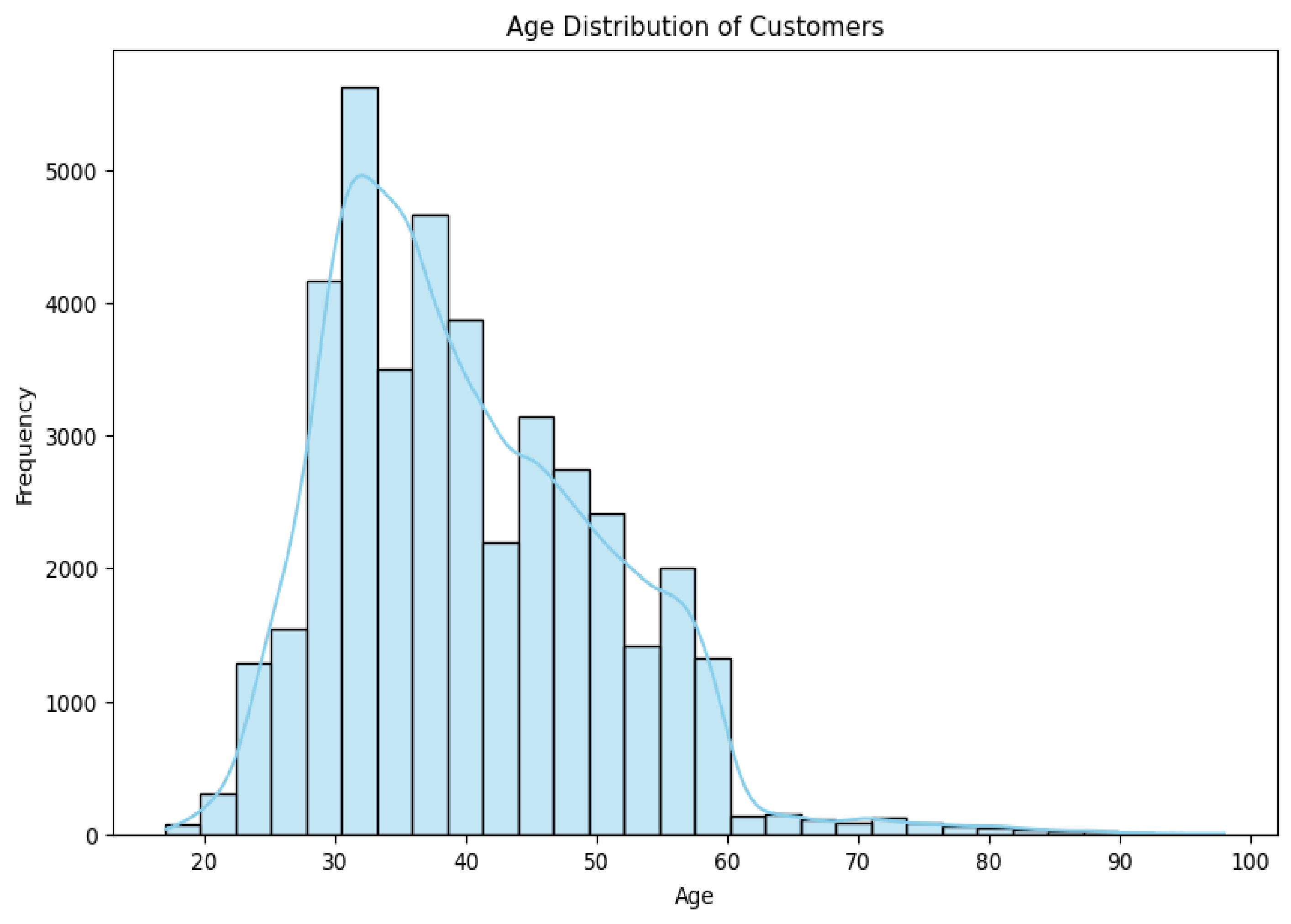
```
# Apply SMOTE to balance the dataset
smote = SMOTE(random_state=42)
X_balanced, y_balanced = smote.fit_resample(X, y)

# Check the distribution after SMOTE
balanced_class_proportion = pd.Series(y_balanced).value_counts(normalize=True) * 100
print(balanced_class_proportion)
```

```
y_encoded
0    50.0
1    50.0
Name: proportion, dtype: float64
```

QUESTION 1

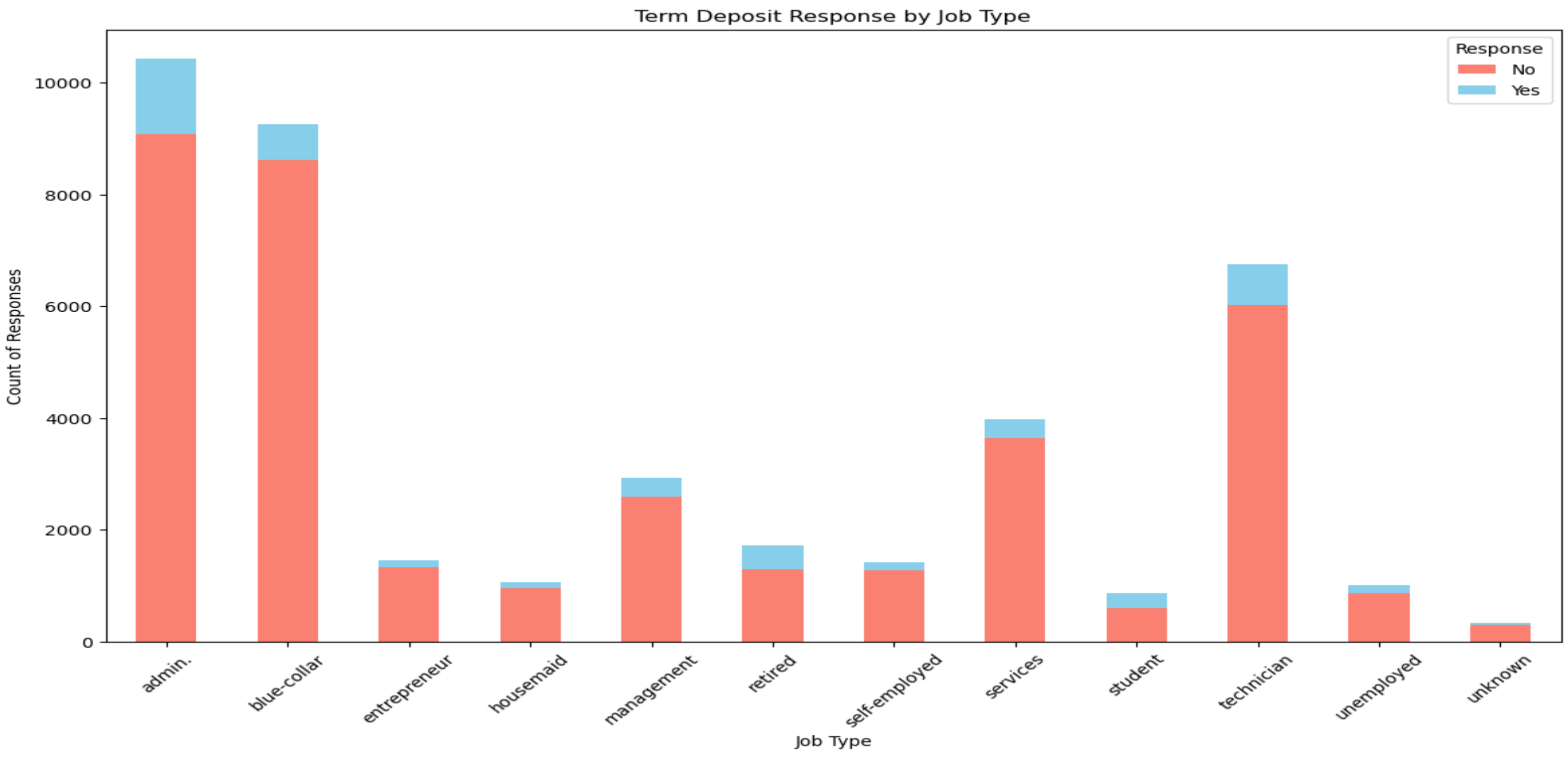
WHAT IS THE AGE DISTRIBUTION OF CUSTOMERS IN THE DATASET?

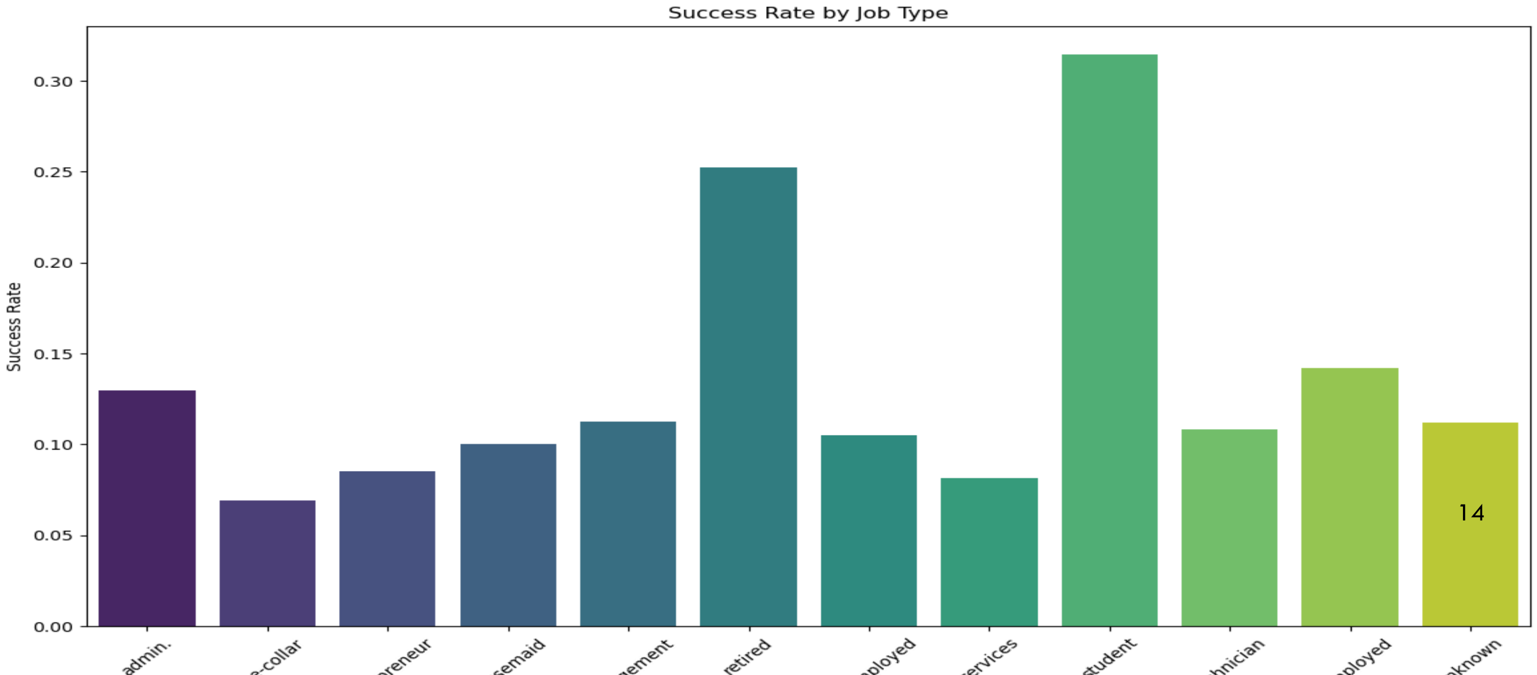
Insight: the customer age distribution is concentrated around young to middle-aged adults, particularly between 25 and 40 years old.

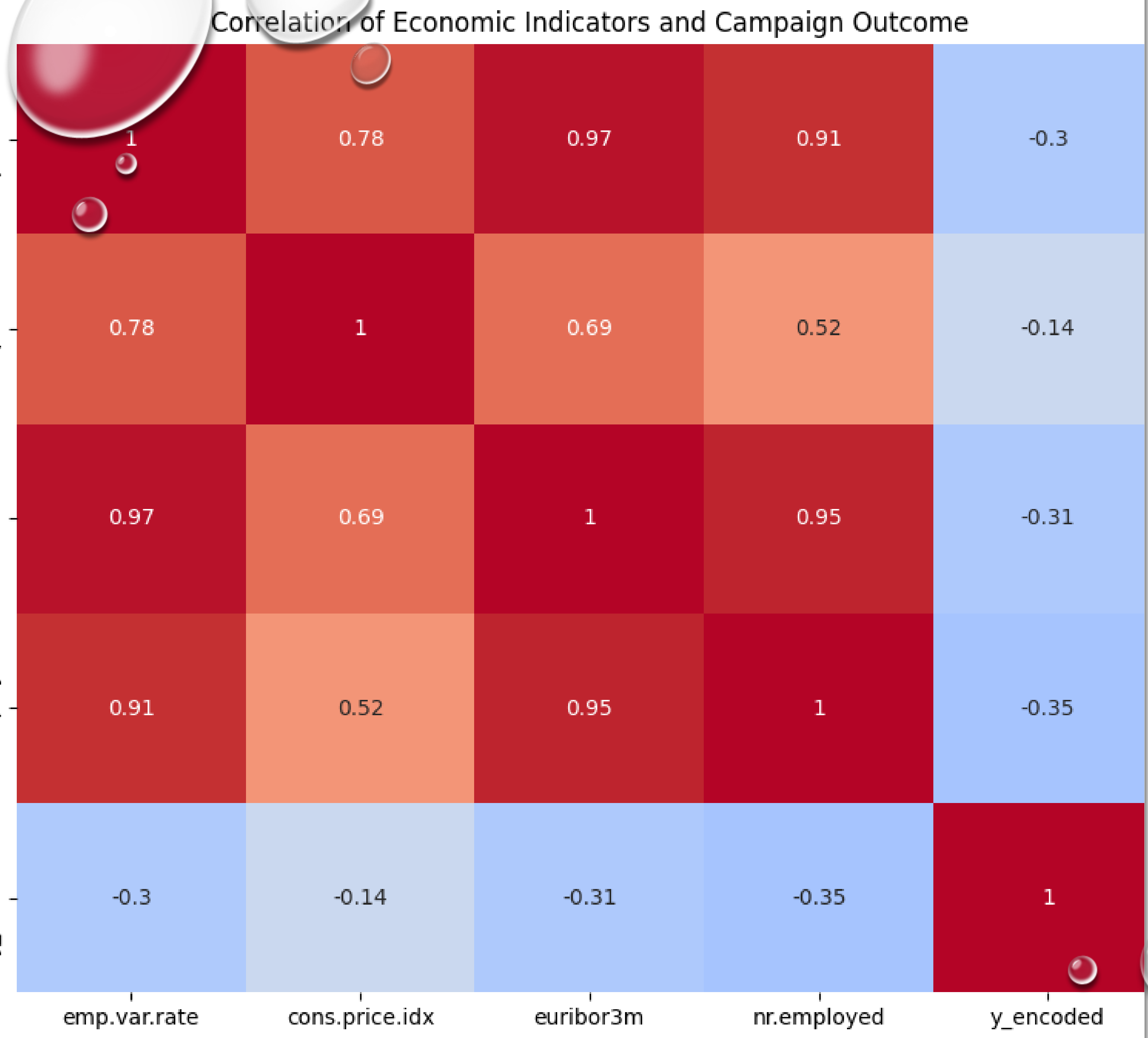


QUESTION 2
WHICH JOB TYPES SHOW THE HIGHEST SUCCESS RATES IN THE MARKETING CAMPAIGN?

	No	Yes	Success Rate
job			
admin.	9070	1352	0.129726
blue-collar	8616	638	0.068943
entrepreneur	1332	124	0.085165
housemaid	954	106	0.100000
management	2596	328	0.112175
retired	1286	434	0.252326
self-employed	1272	149	0.104856
services	3646	323	0.081381
student	600	275	0.314286
technician	6013	730	0.108260
unemployed	870	144	0.142012
unknown	293	37	0.112121





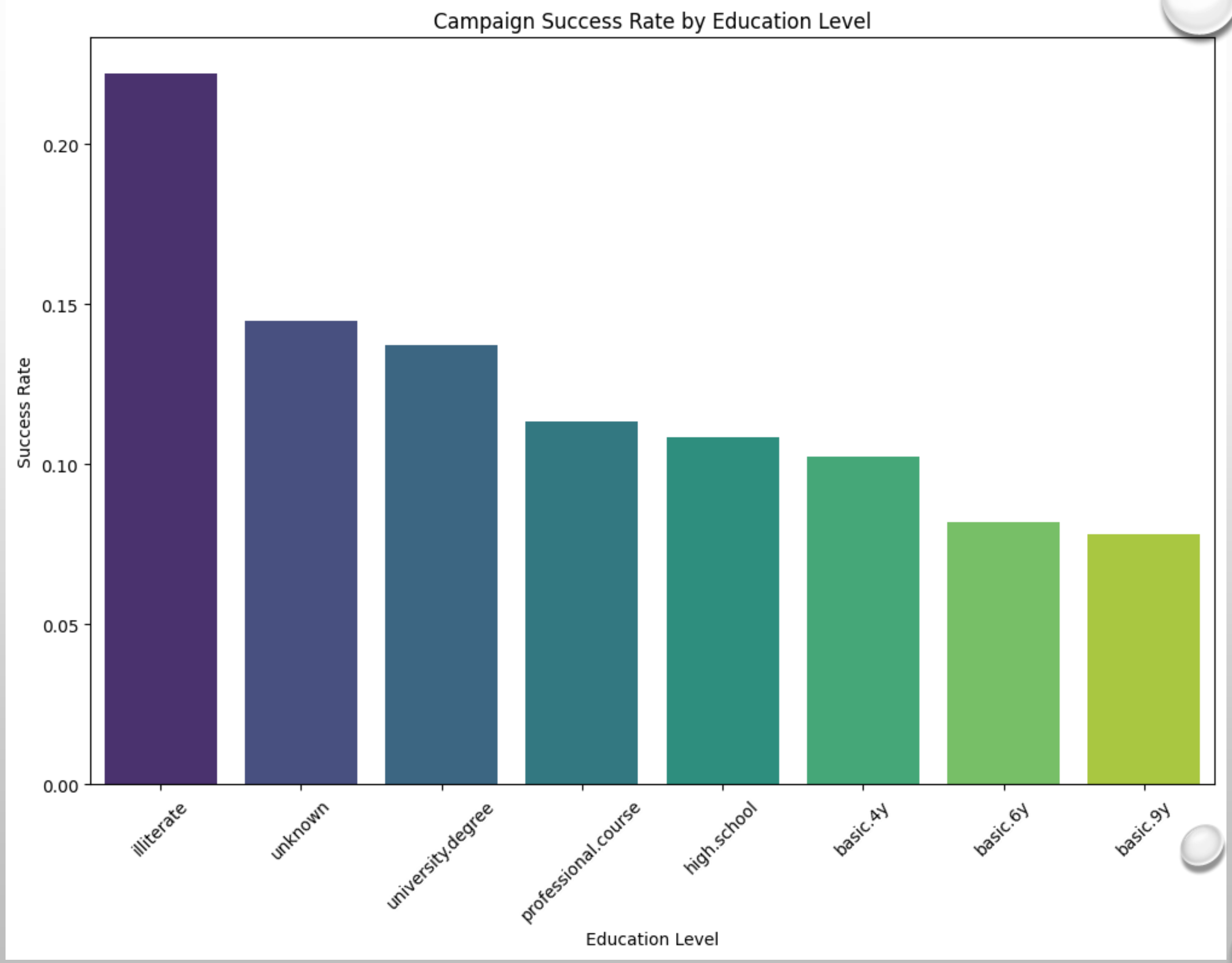


QUESTION 3 HOW DO ECONOMIC INDICATORS IMPACT THE OUTCOMES OF THE MARKETING CAMPAIGN?

Insight: The economic indicators have very weak correlations with the campaign outcomes, suggesting that these factors do not strongly influence whether customers subscribe to the term deposit.

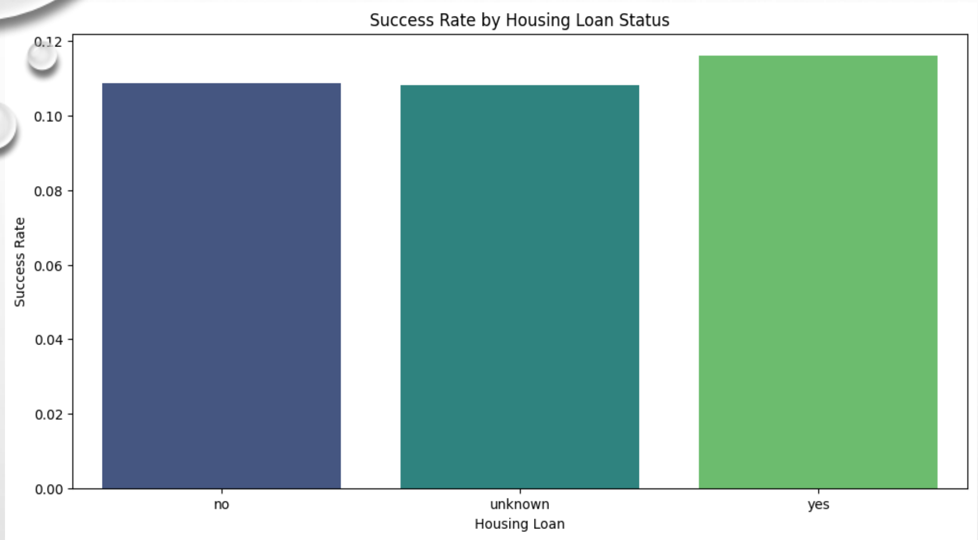
QUESTION 4 IS THERE A SIGNIFICANT VARIATION IN CAMPAIGN SUCCESS ACROSS DIFFERENT EDUCATION LEVELS?

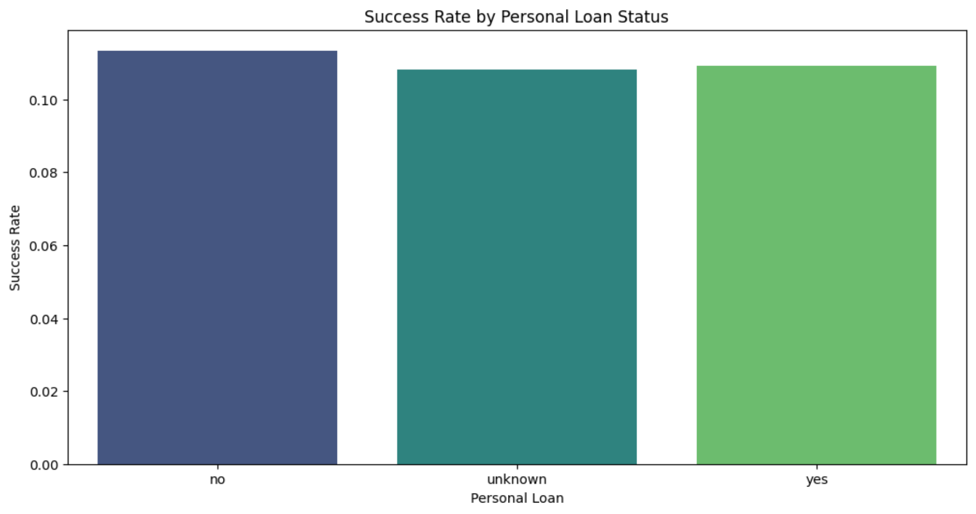
Insight: Illiterate individuals have the highest campaign success rate, followed by those with a university degree. Other education levels have lower success rates, indicating a potential gap in targeting these groups effectively.



QUESTION 5

DOES THE PRESENCE OF A HOUSING OR PERSONAL LOAN AFFECT A CUSTOMER'S DECISION TO SUBSCRIBE TO A TERM DEPOSIT?

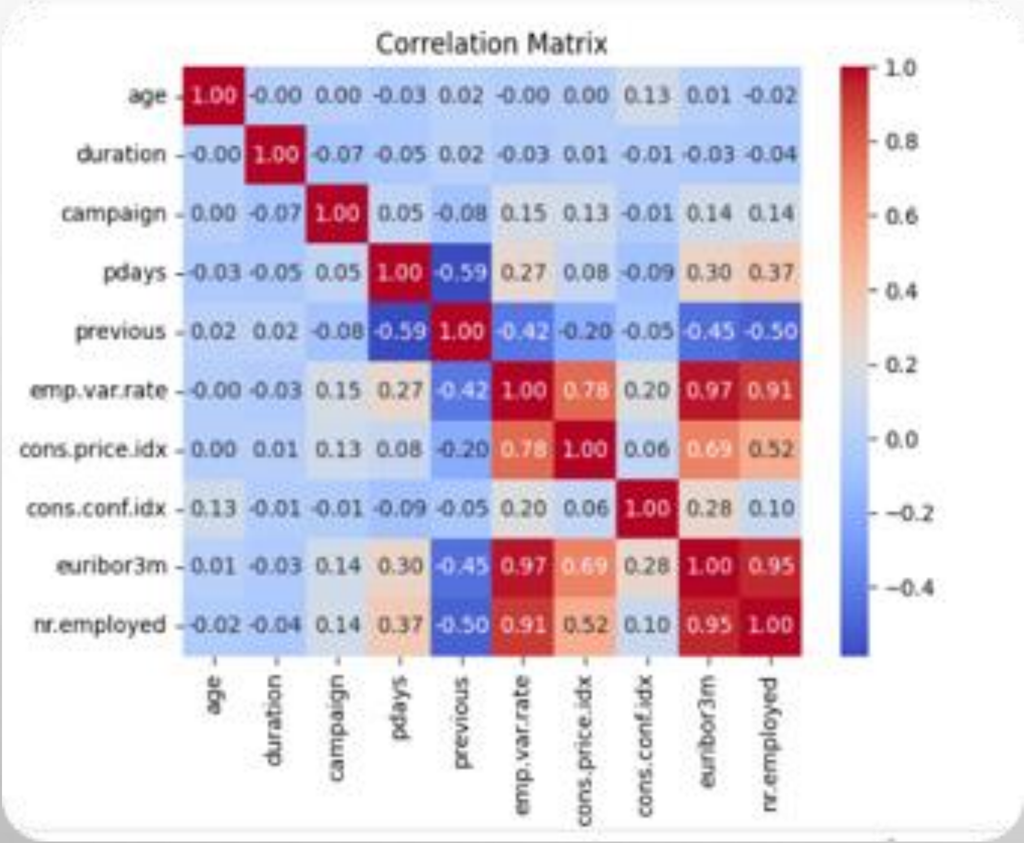




Customers with housing loans are slightly more likely to subscribe to a term deposit, while those with unknown housing loan statuses show the lowest success rates.

PREDICTIVE MODELING

- **LOGISTIC REGRESSION**



	variable	VIF
0	age	16.047296
1	duration	$\boxed{2.011044}$
2	campaign	$\boxed{1.921499}$
3	pdays	44.413175
4	previous	$\boxed{2.001464}$
5	emp.var.rate	28.910219
6	cons.price.idx	22561.123124
7	cons.conf.idx	120.086975
8	euribor3m	226.237349
9	nr.employed	26746.634212

PREDICTIVE MODELING

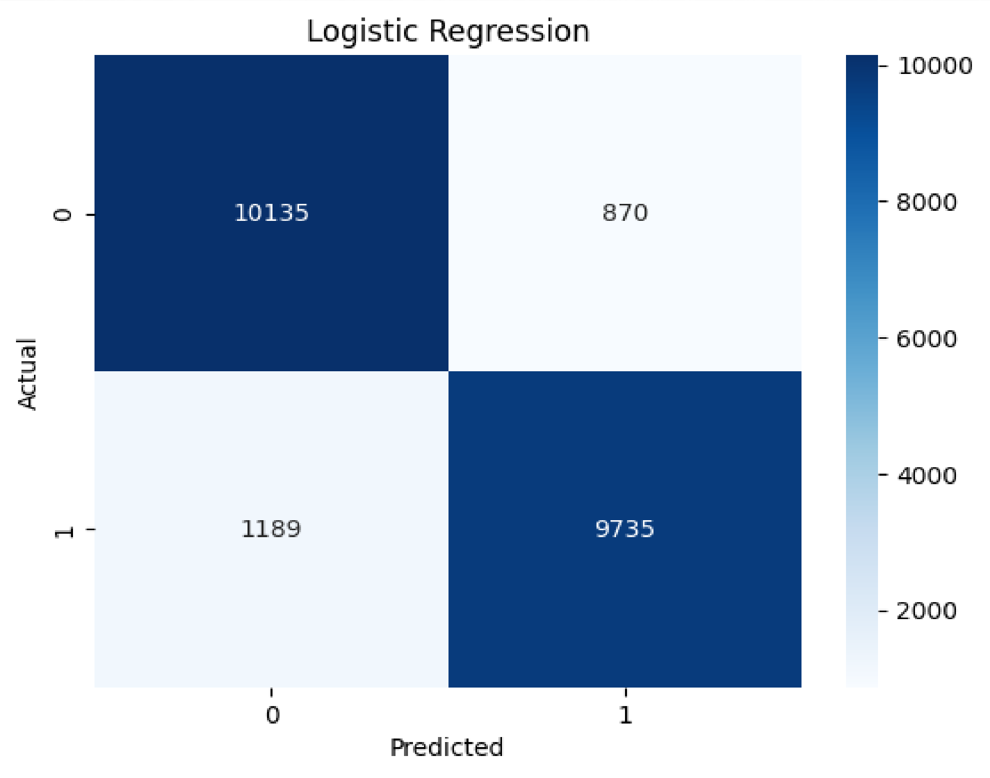
- **LOGISTIC REGRESSION**

```
selected features: Index(['duration', 'campaign', 'previous', 'job_blue-collar', 'job_retired',
       'job_student', 'job_technician', 'marital_married', 'marital_single',
       'education_basic.6y', 'education_basic.9y', 'education_high.school',
       'education_professional.course', 'education_university.degree',
       'education_unknown', 'default_unknown', 'housing_yes', 'loan_yes',
       'contact_telephone', 'month_jun', 'month_mar', 'month_may', 'month_oct',
       'month_sep', 'day_of_week_mon', 'day_of_week_thu', 'day_of_week_tue',
       'day_of_week_wed', 'poutcome_success'],
      dtype='object')
```

```
Lasso Logistic regression coefficients:
[[ 1.57275335 -0.39363192  0.08927271  0.05468542  0.          0.
   0.          1.18396944  0.          0.          0.32814042  0.03901773
   0.          0.          0.75400731  0.89328973  0.          0.01856352
   0.40528481  0.90824975  0.          0.83871999  1.29789019  0.68414577
  -0.03662284  0.          0.          0.48307411  0.          0.04206057
  -0.36141473  0.          0.          0.          0.2659565   1.20807046
  -0.1752708   0.          1.01620986  0.26986051  0.80023776  0.88889439
   0.92246756  0.85452446  0.          1.54043124]]
```

PREDICTIVE MODELING

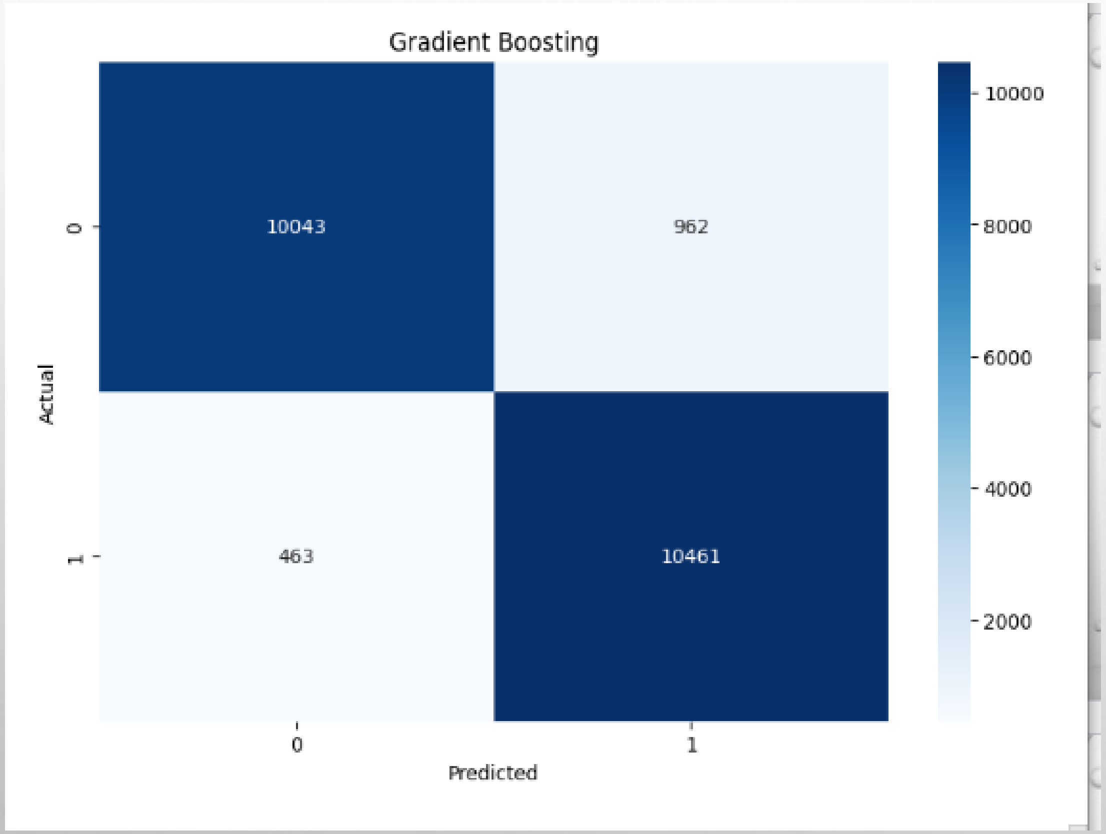
- **LOGISTIC REGRESSION**




```
Accuracy score: 0.9061060695882165
Classification report:
              precision    recall  f1-score   support

           0       0.90      0.92      0.91     11005
           1       0.92      0.89      0.90     10924

    accuracy                           0.91     21929
   macro avg       0.91      0.91      0.91     21929
weighted avg       0.91      0.91      0.91     21929
```

PREDICTIVE MODELING

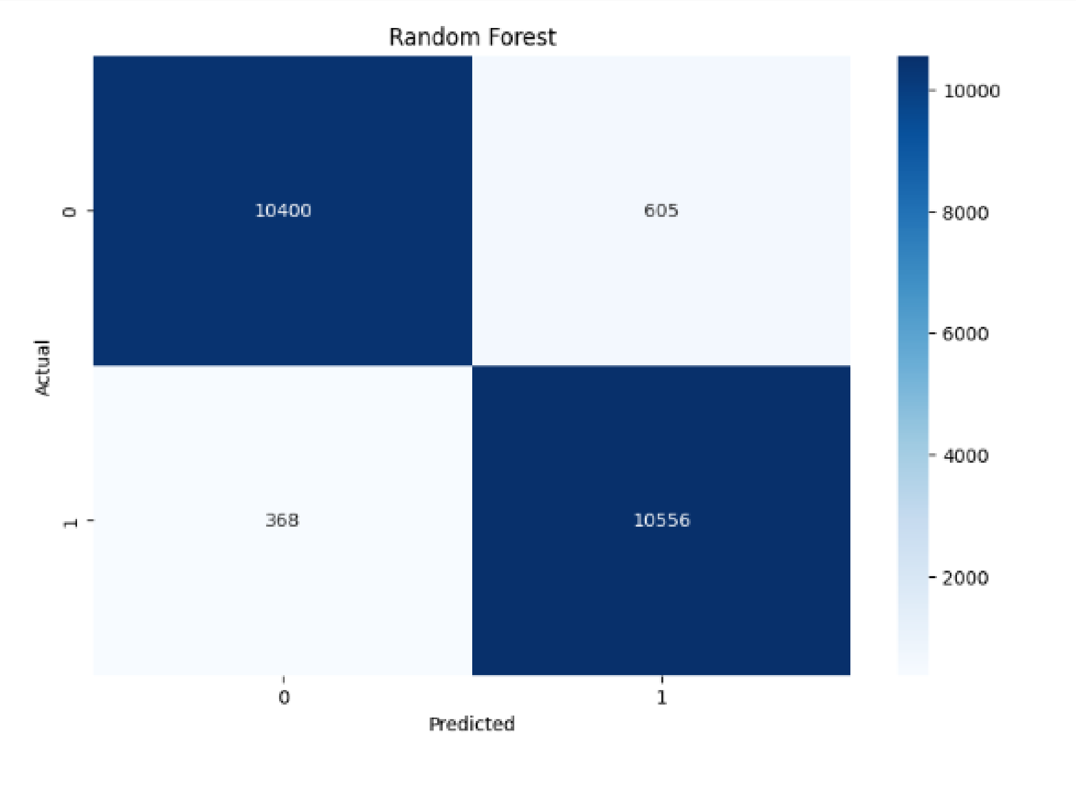
- **GRADIENT BOOSTING**




```
Gradient Boosting Accuracy: 0.9350175566601304
Confusion Matrix:
[[10043   962]
 [  463 10461]]
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.91      0.93     11005
           1       0.92      0.96      0.94     10924

    accuracy                           0.94     21929
   macro avg       0.94      0.94      0.93     21929
weighted avg       0.94      0.94      0.93     21929
```

PREDICTIVE MODELING

- **RANDOM FOREST**




```
Random Forest Accuracy:  0.9556295316703908
Confusion Matrix:
[[10400   605]
 [  368 10556]]
Classification Report:
              precision    recall  f1-score   support

           0       0.97      0.95      0.96     11005
           1       0.95      0.97      0.96     10924

    accuracy                           0.96     21929
   macro avg       0.96      0.96      0.96     21929
weighted avg       0.96      0.96      0.96     21929
```

CONCLUSION

- The Random Forest model outperforms both Logistic Regression and Gradient Boosting, making it the best predictive model among the three.

- The insights from this analysis provide a foundation for ongoing refinement and optimization of marketing strategies

REFERENCES

- HASTIE, T., FRIEDMAN, J., & TIBSHIRANI, R. (2001). *THE ELEMENTS OF STATISTICAL LEARNING : DATA MINING, INFERENCE, AND PREDICTION.*
- SPRINGER.JAMES, G. M., WITTEN, D., HASTIE, T. J., & TIBSHIRANI, R. (2013). *AN INTRODUCTION TO STATISTICAL LEARNING : WITH APPLICATIONS IN R*. SPRINGER.PROVOST, F., & FAWCETT, T. (2013). *DATA SCIENCE FOR BUSINESS : WHAT YOU NEED TO KNOW ABOUT DATA MINING AND DATA-ANALYTIC THINKING.*
- O'REILLY MEDIA.TUKEY, J. W. (1977). *EXPLORATORY DATA ANALYSIS*. ADDISON-WESLEY PUB. CO.