Project: Bank Marketing (Campaign)

Name: Leenar Garnta,Adama Sall,Najma Abdi

Email: najmoa5@gmail.com,leenarganta@gmail.com, adamal.sall@gmail.com,

Country:Kenya,Usa,Saudi Arabia

Batch Code: LISUM45

Specialization: Data Science

Submitted to: Data Glacier (Group project)

Table of Contents

1. Problem Description

ABC Bank is planning to launch a new term deposit product.Before investing in a large-scale marketing campaign,the bank wants to use machine learning model to identify customers who are highly likely to subscribe ,based on their previous interactions with marketing campaigns.

2. Business Understanding

Predicting marketing campaign outcomes and identifying key influencing features helps organizations run more efficient campaigns. By categorizing customers likely to subscribe to a term deposit, businesses can target the right audience using focused marketing channels (telemarketing, SMS, email, etc.), saving time and resources while improving conversion rates.

3. Data understanding

- Types of data

The dataset contains 20 independent variables and 1 dependent variable. The independent variables are the following:

1 - age (numeric)
2 - job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
4 - education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5 - default: has credit in default? (categorical: 'no','yes','unknown')

6 - housing: has housing loan? (categorical: 'no','yes','unknown')
7 - loan: has personal loan? (categorical: 'no','yes','unknown')
related with the last contact of the current campaign:
8 - contact: contact communication type (categorical: 'cellular','telephone')
9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')
10 - day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
11 - duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.
12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
14 - previous: number of contacts performed before this campaign and for this client (numeric)
15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')
social and economic context attributes
16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)
17 - cons.price.idx: consumer price index - monthly indicator (numeric)
18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)
19 - euribor3m: euribor 3 month rate - daily indicator (numeric)
20 - nr.employed: number of employees - quarterly indicator (numeric)
Output variable (desired target):
21 - y - has the client subscribed a term deposit? (binary: 'yes','no')

The independent variables are a mix of categorical and numerical values.

As instructed in the dataset, the **duration** feature is dropped from the dataset.

- Problems in the data

  The dataset contains a total of 41188 rows, out of which there were a total of 12 duplicate rows. The pandas **drop_duplicates** method is used to drop the duplicate rows.

Missing Values:

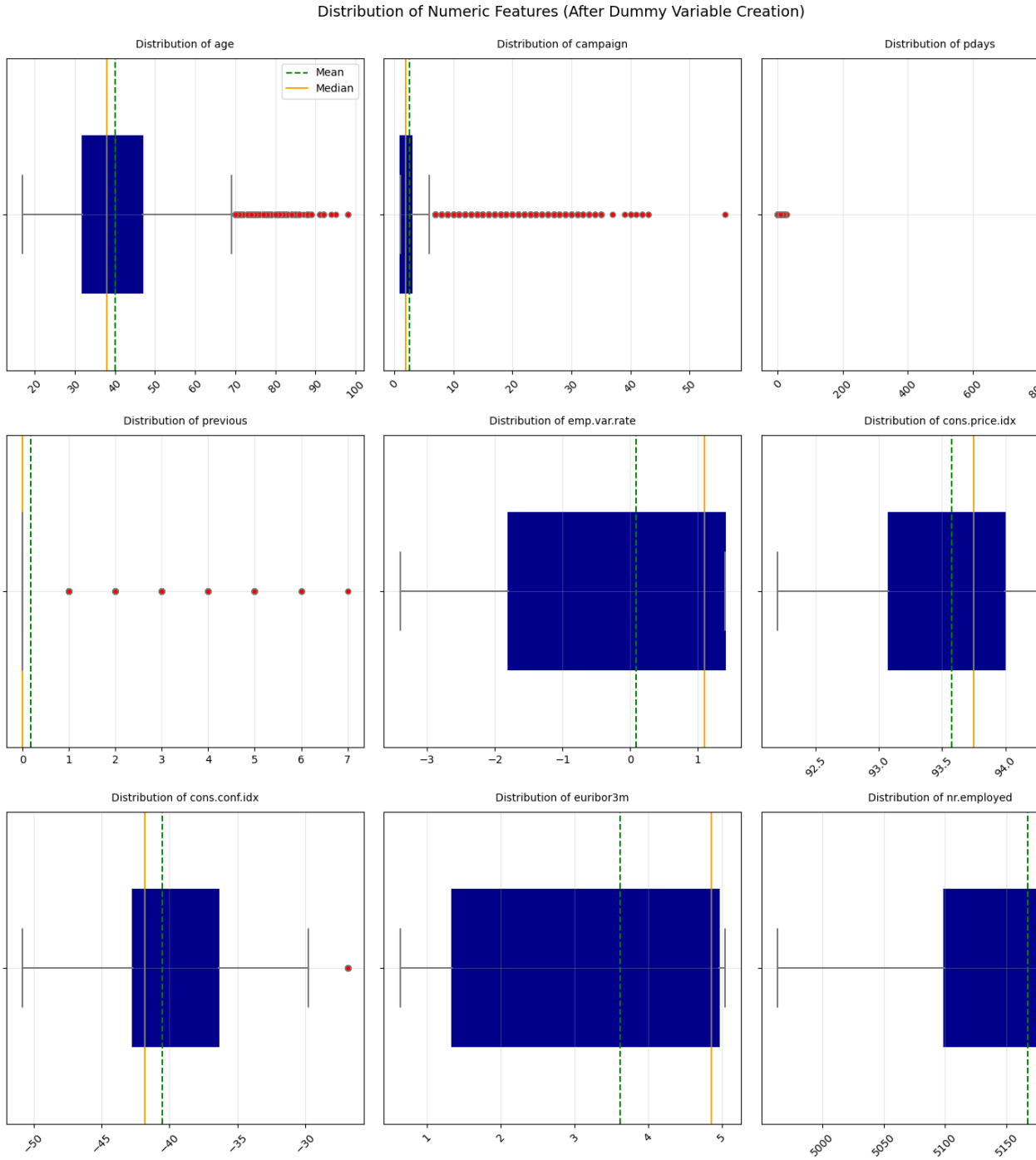
In [2]:
```
# Check for 'unknown' values and NA
print("Missing values:")
print(data.isna().sum())
print("\nUnknown values per column:")
print((data == 'unknown').sum())
```

```
Missing 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
nr.employed       0
y                 0
dtype: int64
```

There are no missing values in the dataset.

Outlier detection

Outliers are detected using box plots.



Outliers represent those values which are at an abnormal distance from the central distribution of data points. These values affect the statistics of the data mainly mean and mode. Therefore it's important to deal with outliers in the data cleaning stage so that the performance of the model don't get compromised.

In the box plot, outliers are seen in the features **age, campaign, pdays and previous** which are indicated as data points outside the whiskers of the box plot.

- Approaches to overcome the problems

  Considering the statistics of the outlier features

```
In [16]: #checking statistics of outlier features

         data[['age', 'pdays', 'campaign', 'previous']].describe()
```

Out[16]:

	age	pdays	campaign	previous
count	41176.00000	41176.000000	41176.000000	41176.000000
mean	40.02380	962.464810	2.567879	0.173013
std	10.42068	186.937102	2.770318	0.494964
min	17.00000	0.000000	1.000000	0.000000
25%	32.00000	999.000000	1.000000	0.000000
50%	38.00000	999.000000	2.000000	0.000000
75%	47.00000	999.000000	3.000000	0.000000
max	98.00000	999.000000	56.000000	7.000000

i. **age:** The maximum value of age is 98 which seems to be realistic. Therefore, it's not dropped.

ii. **pdays:** number of days that passed by after the client was last contacted from a previous campaign. From the statistics, the maximum value of **pdays** is 999. In the features description, this value means the client was not previously contacted. Moreover around 96% of rows contains this value, therefore dropping rows containing 999 seems unrealistic.

```
In [17]: len(data[data['pdays'] == 999]) / len(data) * 100

Out[17]: 96.32067223625413
```

iii. **campaign:** 'campaign' holds the number of contacts performed during this campaign and for this client. From the statistics, the maximum value is given as 56 which is clearly noise. Numbers for 'campaign' above 20 is around 0.38%.

```
In [20]: len(data[data['campaign'] > 20]) / len(data) * 100

Out[20]: 0.38129007188653585
```

Therefore, I suggest to impute those rows with average of campaign values.

iv. **previous:** 'previous' holds the number of contacts performed before this campaign and for this client. Since the maximum value of 7 doesn't seem to be a noise, we I chose to ignore this outlier.

4. Treating outliers

1.Strategic Capping

In [23]:
```
# Remove customers who were contacted more than 7 times
capped_data = data[data['campaign'] <= 7]

# Recalculate overall conversion under the capped policy
capped_overall_conv = capped_data['y'].mean()
print(f"Overall conversion after capping at 7: {capped_overall_conv*100:.2f}%")
```

```
Overall conversion after capping at 7: 11.59%
```

After strategic capping,the maximum value has now become 7

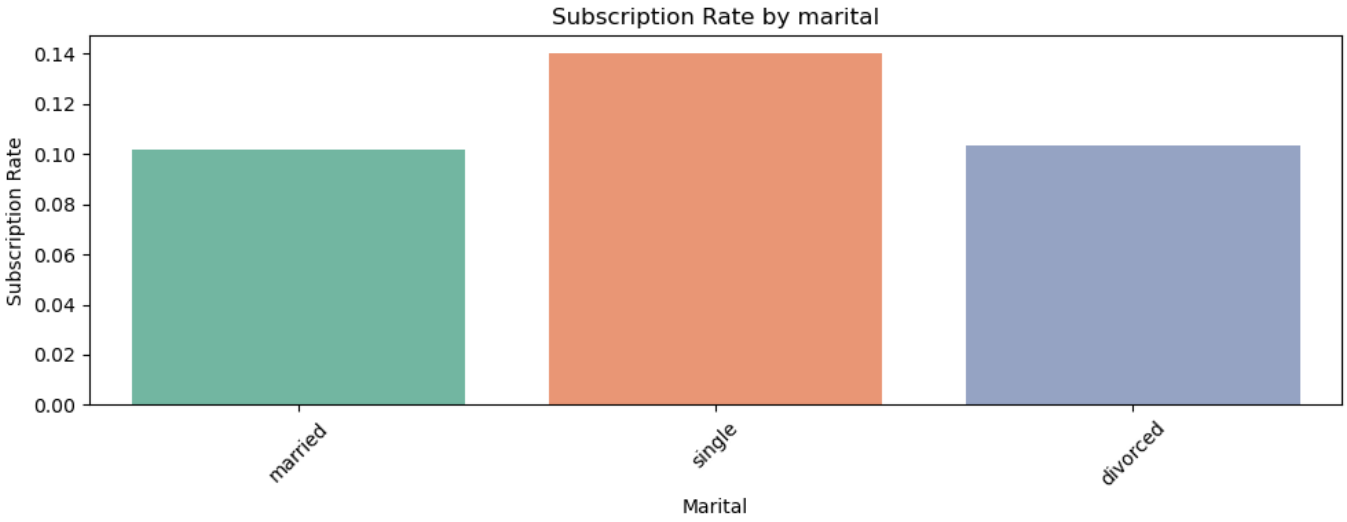
2.Winsorization

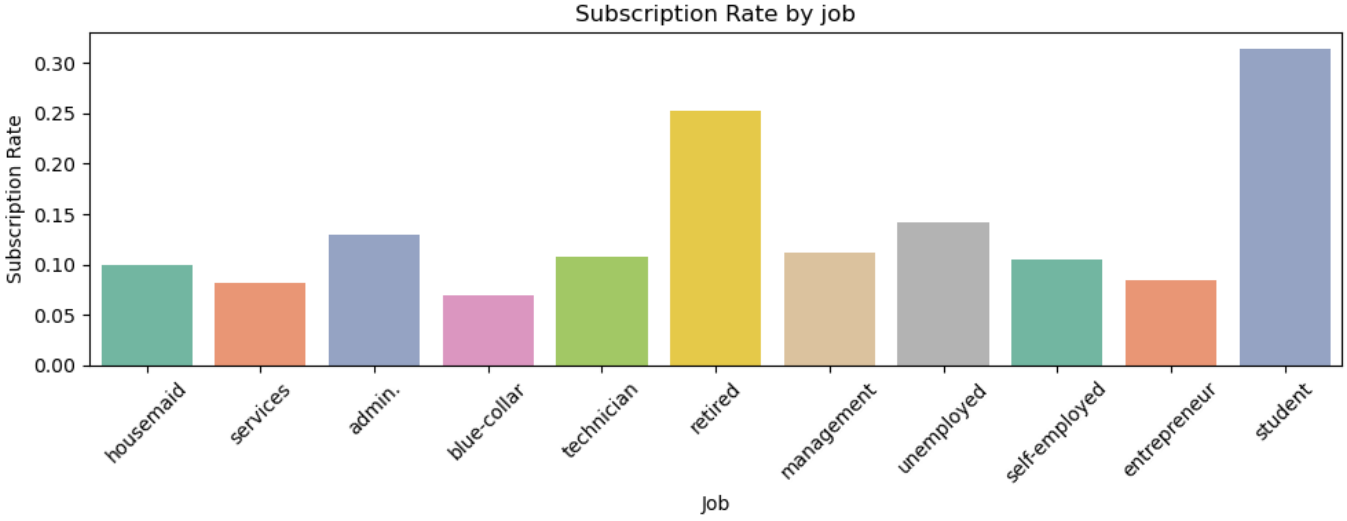
In [24]:
```
# Set your thresholds (e.g., 1st and 99th percentiles)
lower_limit = data['campaign'].quantile(0.01)
upper_limit = data['campaign'].quantile(0.99)

# Winsorize: Replace extreme values with thresholds
data['campaign_winsorized'] = np.clip(data['campaign'], lower_limit, upper_limit)

# Optional: Compare before and after
print(f"Original Max: {data['campaign'].max()} → Winsorized Max: {data['campaign_winsorized'].max()}")
```

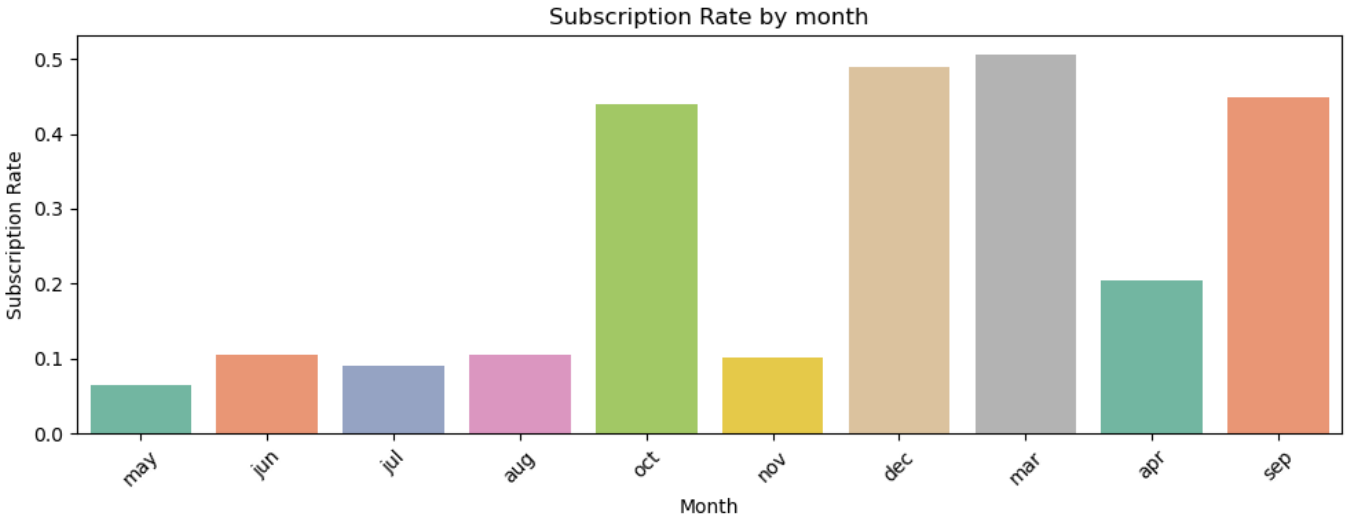
```
Original Max: 56 → Winsorized Max: 14
```

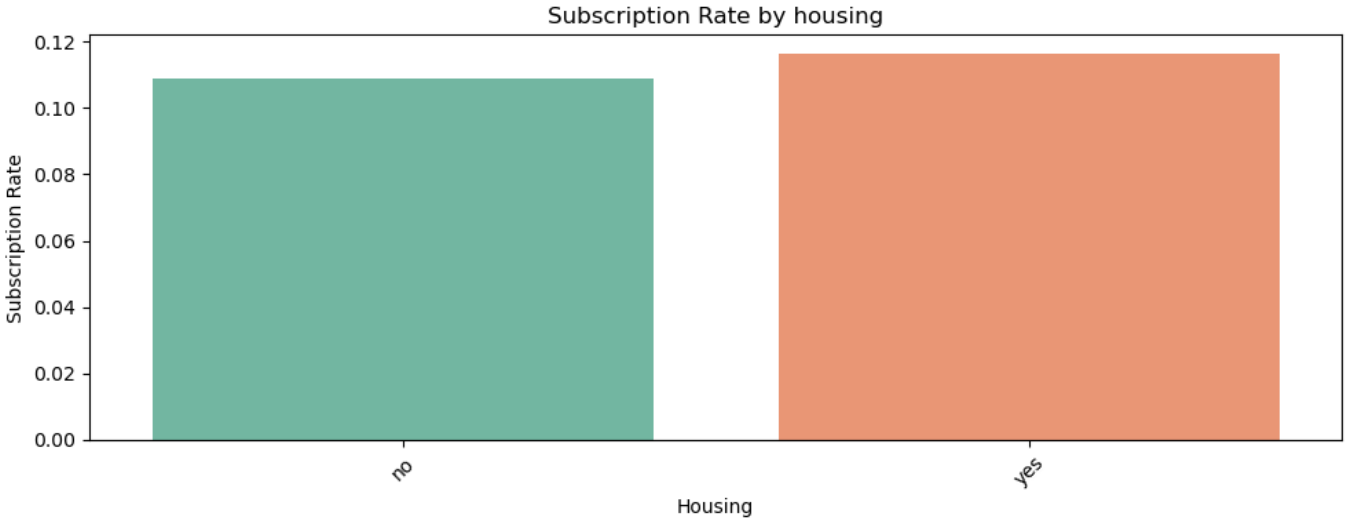
5. EDA and recommendation

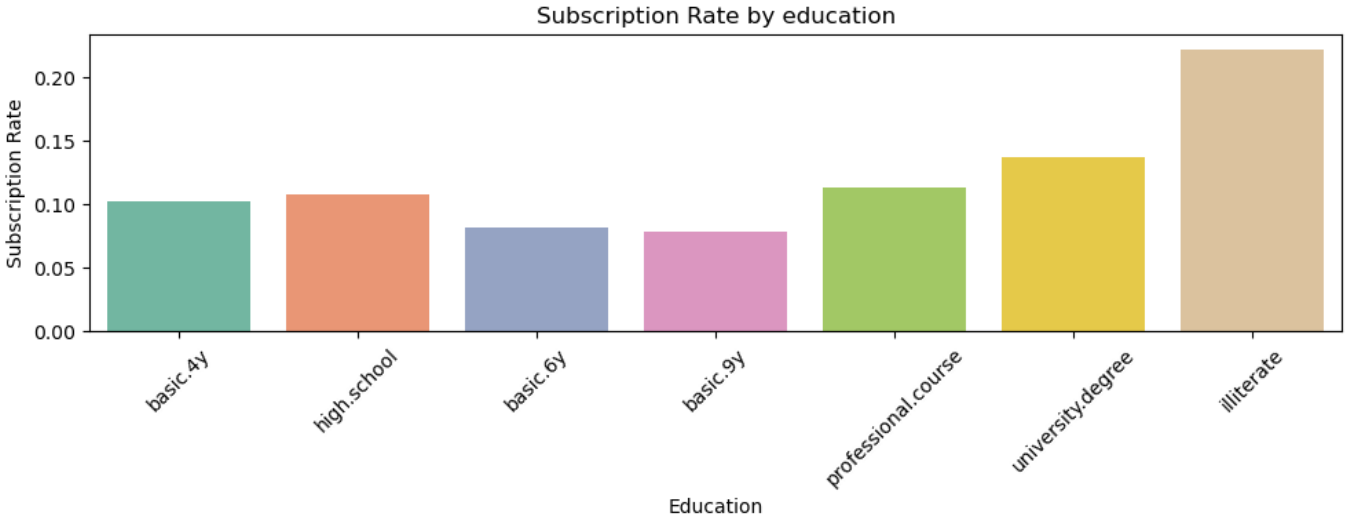
- Exploring categorical values

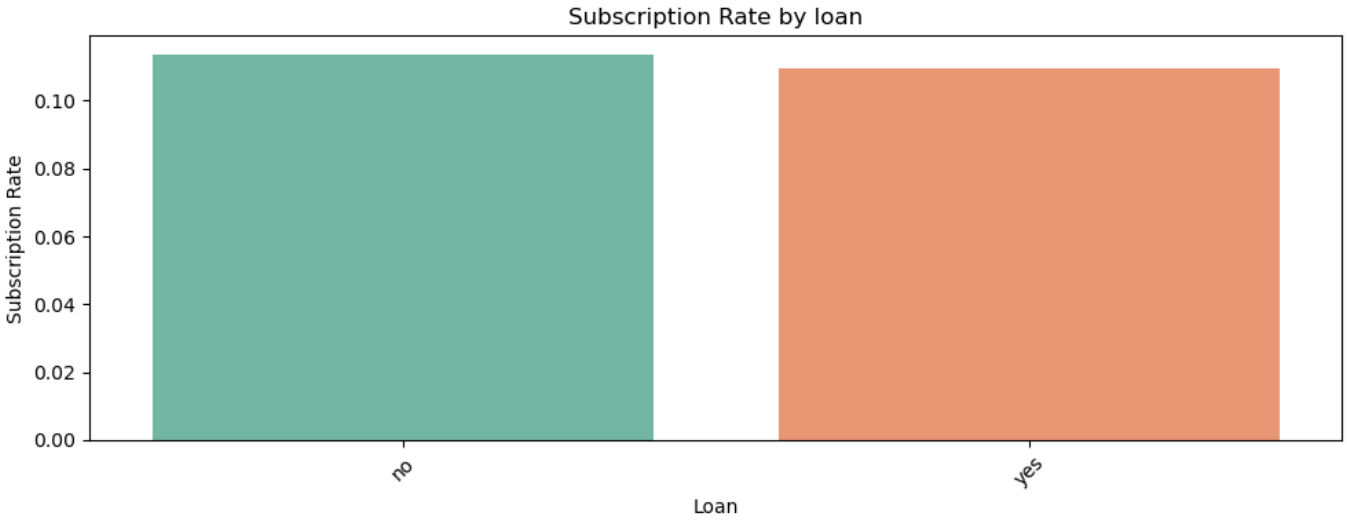




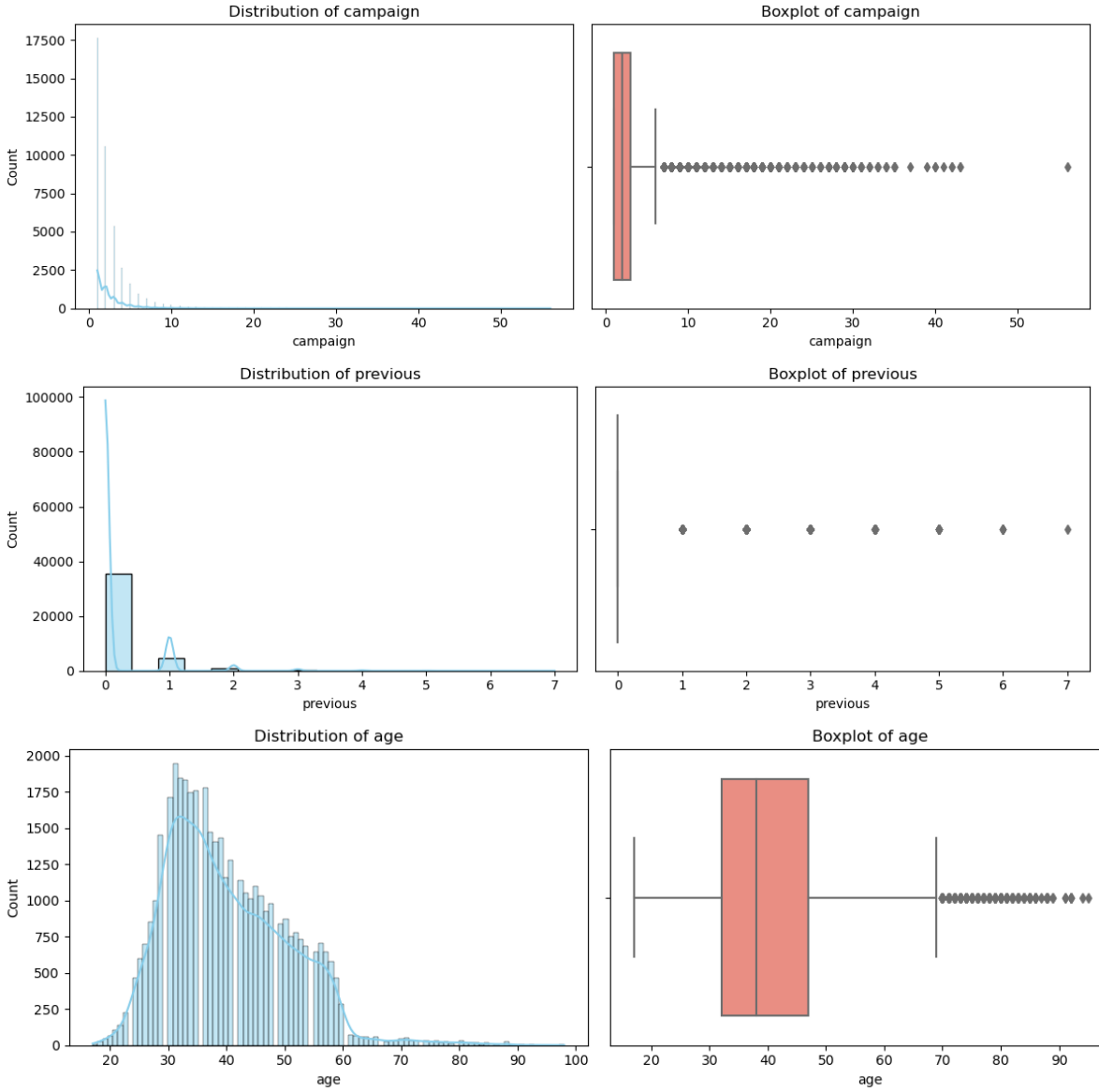

Subscription Rate by month
Subscription Rate
0.0
0.1
0.2
0.3
0.4
0.5
may
jun
jul
aug
oct
nov
dec
mar
apr
sep
Month

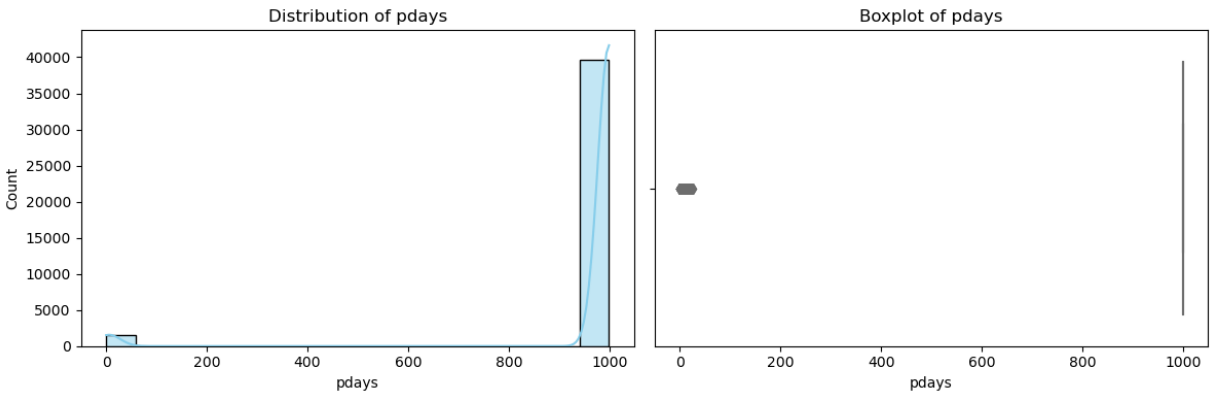

Subscription Rate by housing
Subscription Rate
0.00
0.02
0.04
0.06
0.08
0.10
0.12
no
yes
Housing


Subscription Rate by education
Subscription Rate
0.00
0.05
0.10
0.15
0.20
basic.4y
high.school
basic.6y
basic.9y
professional.course
university.degree
illiterate
Education

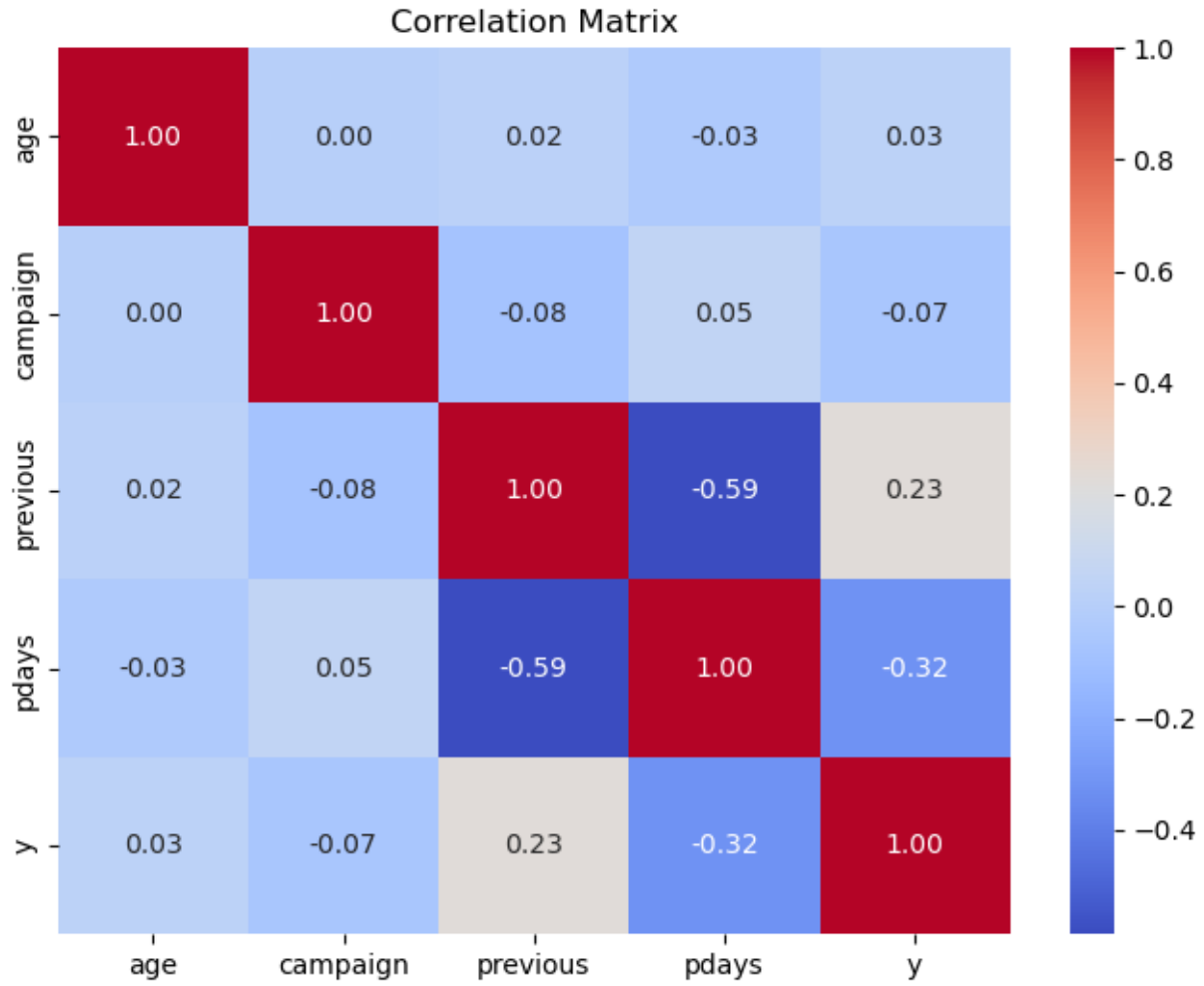

Subscription Rate by loan
Subscription Rate
0.00
0.02
0.04
0.06
0.08
0.10
no
yes
Loan

- Exploring numerical features

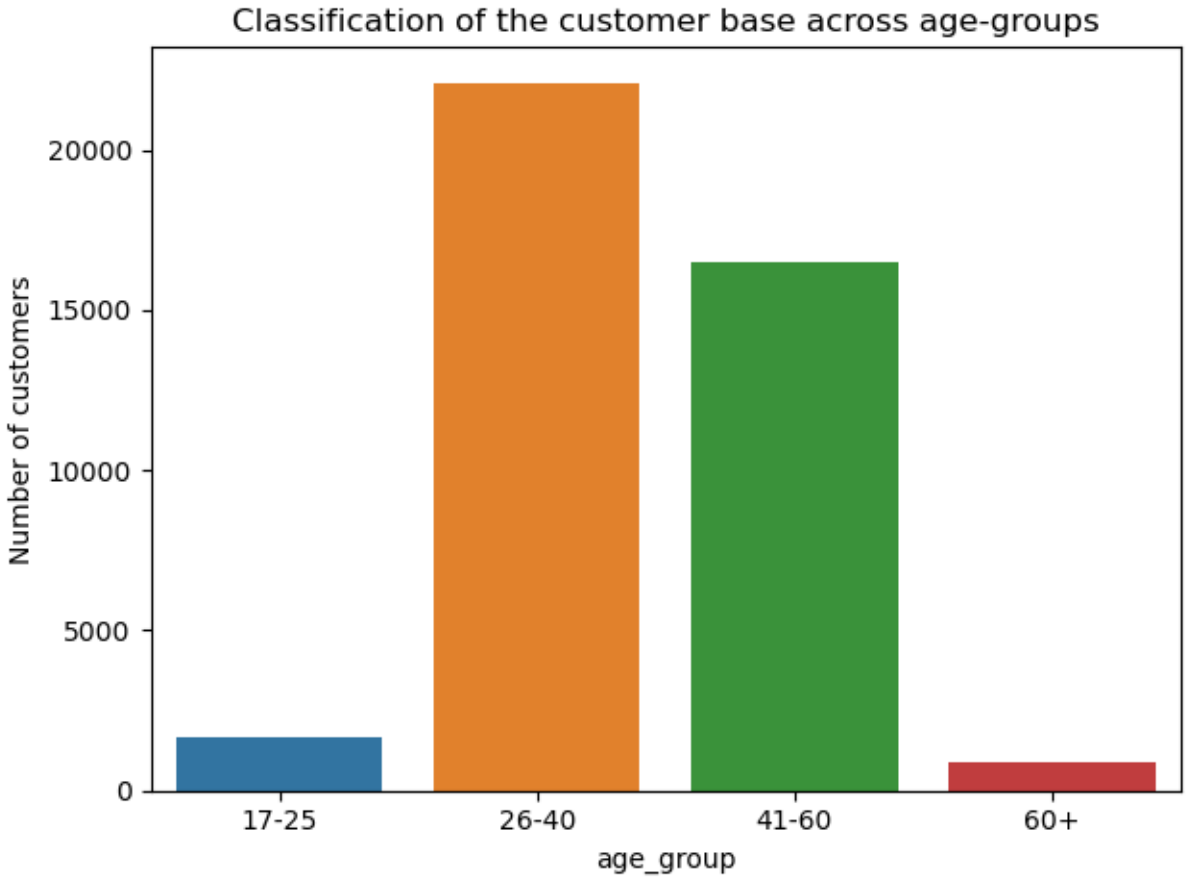



Distribution of pdays
Count
40000
35000
30000
25000
20000
15000
10000
5000
0
0
200
400
600
800
1000
pdays
Boxplot of pdays
0
200
400
600
800
1000
pdays

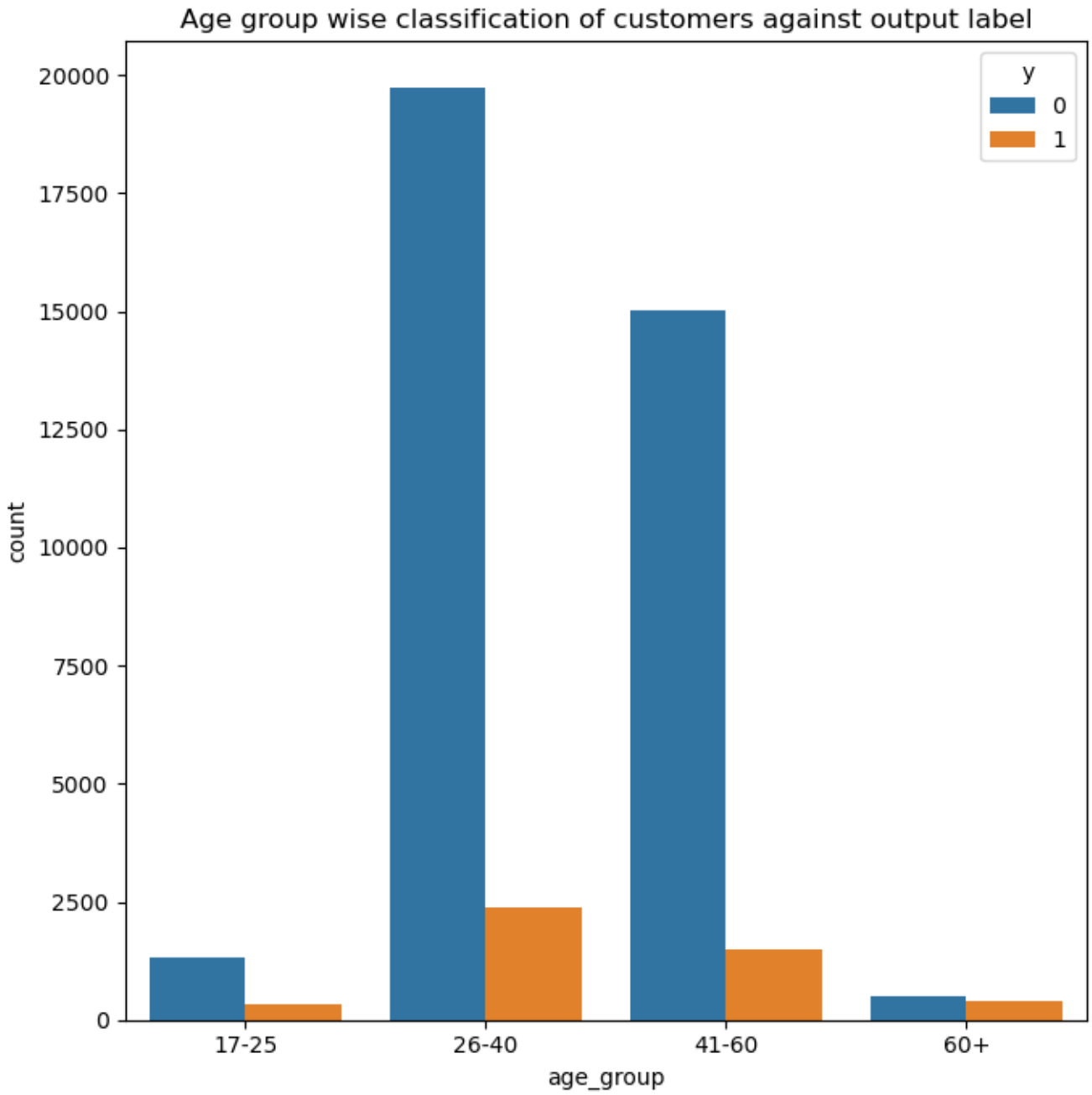
- Correlation Matrix



- Classification of the customer base across age-groups

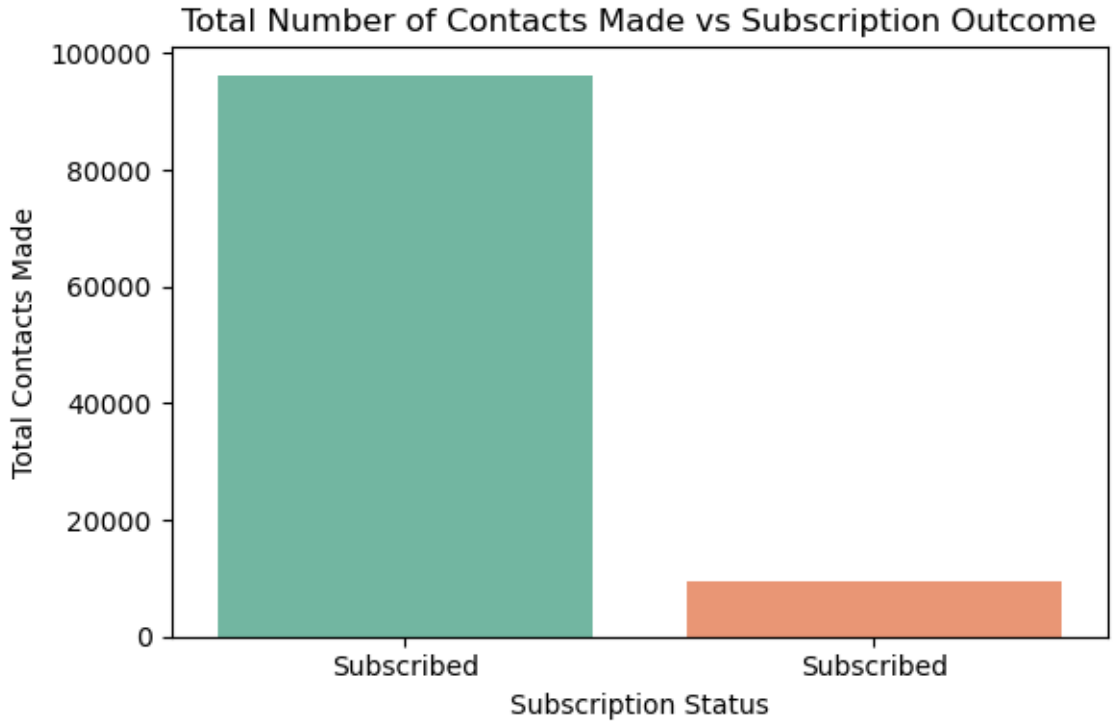


- Looking at relation between different age groups and the output label y

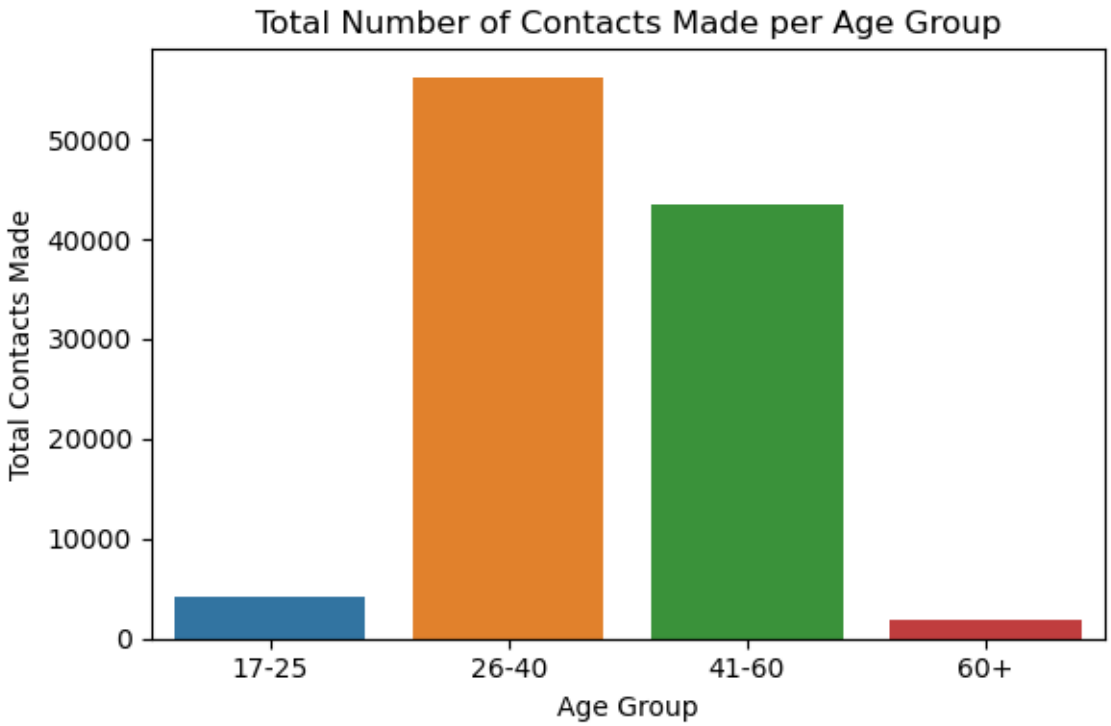

Age group wise classification of customers against output label
y
0
1
20000
17500
15000
12500
10000
7500
5000
2500
0
count
17-25
26-40
41-60
60+
age_group

Observation: In the age-groups of 26-40 and 41-60 yrs, majority of the people are not subscribed to the term deposit plan

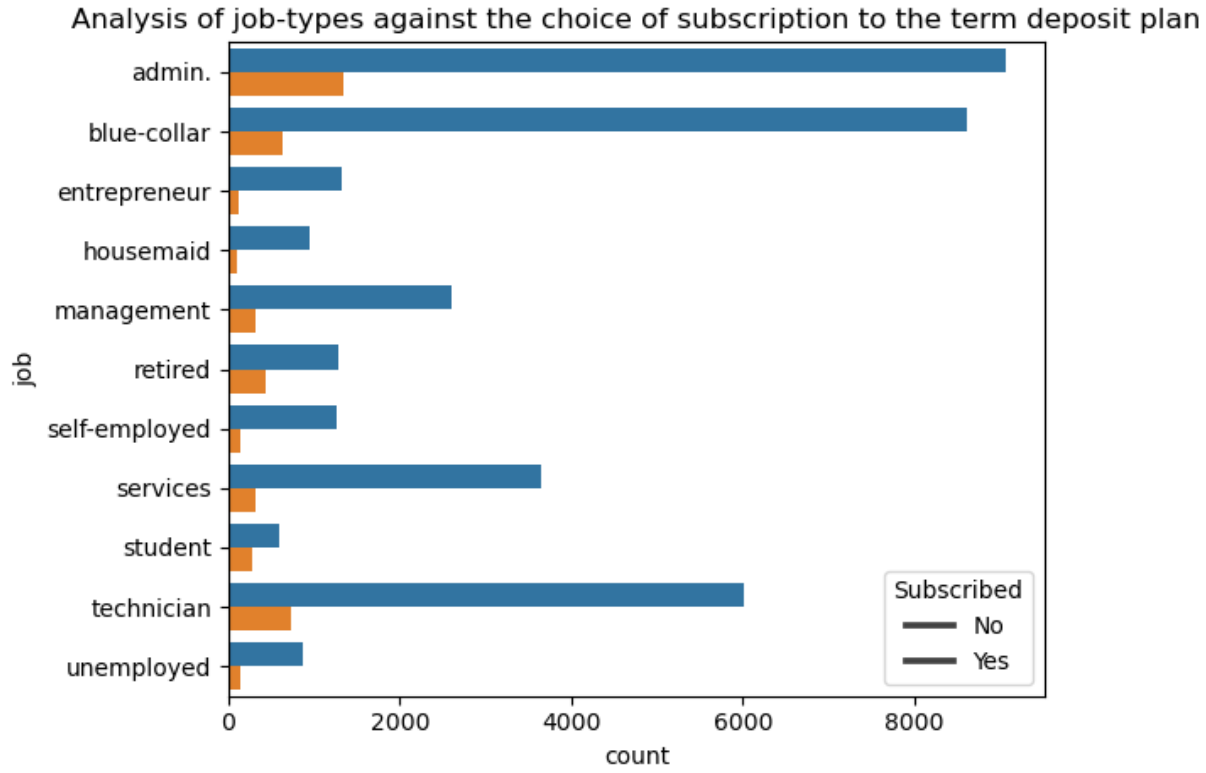
- Looking at relation between Number of contacts made vs subscription outcome



Observation: When a greater number of contacts is made to the customer, they haven't subscribed to the term deposit plan



The 26-40 and 41-60 age-groups witness majority of the contacts made in this campaign. These two age-groups seem to be the target groups for the bank.



Observation: Looking at the jobs, 'admin', 'blue-collar' and 'technician' are the prominent jobs and most of the customers in these jobs have rejected the term deposit plan.

6. Model Building

Since Machine learning models accepts input features as numbers, all the categorical features are converted to numeric and in the end feature scaling is performed on all features to achieve global minimum fast in gradient descent.

The categorical features include 'job', 'marital', 'education', 'default', 'housing', 'loan', 'contact', 'month', 'day_of_week', 'poutcome', 'campaign' and 'y'.

In [21]:
```
columns =[ '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', 'contacted_before', 'pdays_clean',
       'previous_capped', 'campaign_winsorized']
for col in columns:
    print(col, df[col].unique())
```

```
age [56 57 37 40 45 59 41 24 25 29 35 54 46 50 39 30 55 49 34 52 58 32 38 44
 42 60 53 47 51 48 33 31 43 36 28 27 26 22 23 20 21 61 19 18 70 66 76 67
 73 88 95 77 68 75 63 80 62 65 72 82 64 71 69 78 85 79 83 81 74 17 87 91
 86 98 94 84 92 89]
job ['housemaid' 'services' 'admin.' 'blue-collar' 'technician' 'retired'
 'management' 'unemployed' 'self-employed' nan 'entrepreneur' 'student']
marital ['married' 'single' 'divorced' nan]
education ['basic.4y' 'high.school' 'basic.6y' 'basic.9y' 'professional.course' nan
 'university.degree' 'illiterate']
default ['no' nan 'yes']
housing ['no' 'yes' nan]
loan ['no' 'yes' nan]
contact ['telephone' 'cellular']
month ['may' 'jun' 'jul' 'aug' 'oct' 'nov' 'dec' 'mar' 'apr' 'sep']
day_of_week ['mon' 'tue' 'wed' 'thu' 'fri']
campaign [ 1  2  3  4  5  6  7  8  9 10 11 12 13 19 18 23 14 22 25 16 17 15 20 56
 39 35 42 28 26 27 32 21 24 29 31 30 41 37 40 33 34 43]
pdays [999   6   4   3   5   1   0  10   7   8   9  11   2  12  13  14  15  16
  21  17  18  22  25  26  19  27  20]
previous [0 1 2 3 4 5 6 7]
poutcome ['nonexistent' 'failure' 'success']
emp.var.rate [ 1.1  1.4 -0.1 -0.2 -1.8 -2.9 -3.4 -3.  -1.7 -1.1]
cons.price.idx [93.994 94.465 93.918 93.444 93.798 93.2   92.756 92.843 93.075 92.893
 92.963 92.469 92.201 92.379 92.431 92.649 92.713 93.369 93.749 93.876
```

To perform one hot encoding, dummy variables are created for the features job, marital, education, contact and poutcome.

In [12]:
```
#Convert categorical variables to dummy variables
cat_cols = data.select_dtypes(include='object').columns
data = pd.get_dummies(data, columns=cat_cols, drop_first=True)
```

Training using Machine Learning classifiers

Data Preprocessing:
The dataset was split into features (`X`) and the target variable (`y`). To handle different data types, the features were divided into **categorical** and **numerical** columns.

-

Categorical features were processed using a pipeline that applied:

SimpleImputer: Replaced missing values with the constant "missing".

neHotEncoder: Converted categorical variables into numerical format while ignoring unknown categories

Numerical features were processed using a pipeline that applied:

SimpleImputer: Replaced missing values with the mean of each column.

Both pipelines were combined using a **ColumnTransformer** to ensure the transformations were applied correctly to the respective columns.

Finally, the data was split into **training (70%)** and **testing (30%)** sets using stratified sampling to preserve the class distribution of the target variable.

From all the above Models, **Logistic Regression** performed the best with an accuracy of **90%.**

Model	Accuracy
Logistic Regression	**90%**
Decision Tree	84.13%
Naïve Bayes	80.28%
Random Forest	89.21%
Gradient Boosting	90%

Hyperparameter Tuning

Since Gradient Boosting gave better performance, hyper parameter tuning was performed on it to identify best features.

Parameter	Values
classifier__n_estimators	100, 200
classifier__learning_rate	0.01,0.1,0.2
classifier__max_depth	3,5,7
classifier__subsample	0.8,1.0

```
# Gradient Boosting parameter grid
gb_param_grid = {
    'classifier__n_estimators': [100, 200],
    'classifier__learning_rate': [0.01, 0.1, 0.2],
    'classifier__max_depth': [3, 5, 7],
    'classifier__subsample': [0.8, 1.0]
}
```

```
Best GB Params: {'classifier__subsample': 0.8, 'classifier__n_estimators': 100, 'classifier__max_depth': 3, 'classifier__lea
rning_rate': 0.1}
```

The best parameters values were identified as

Parameter	Values
classifier__n_estimators	100
classifier__learning_rate	0.1
classifier__max_depth	3
classifier__subsample	0.8

With Hyperparameter tuning the accuracy increased from 90**% to 90%.**

Evaluation metrices

The classification report and the confusion matrix are reported as

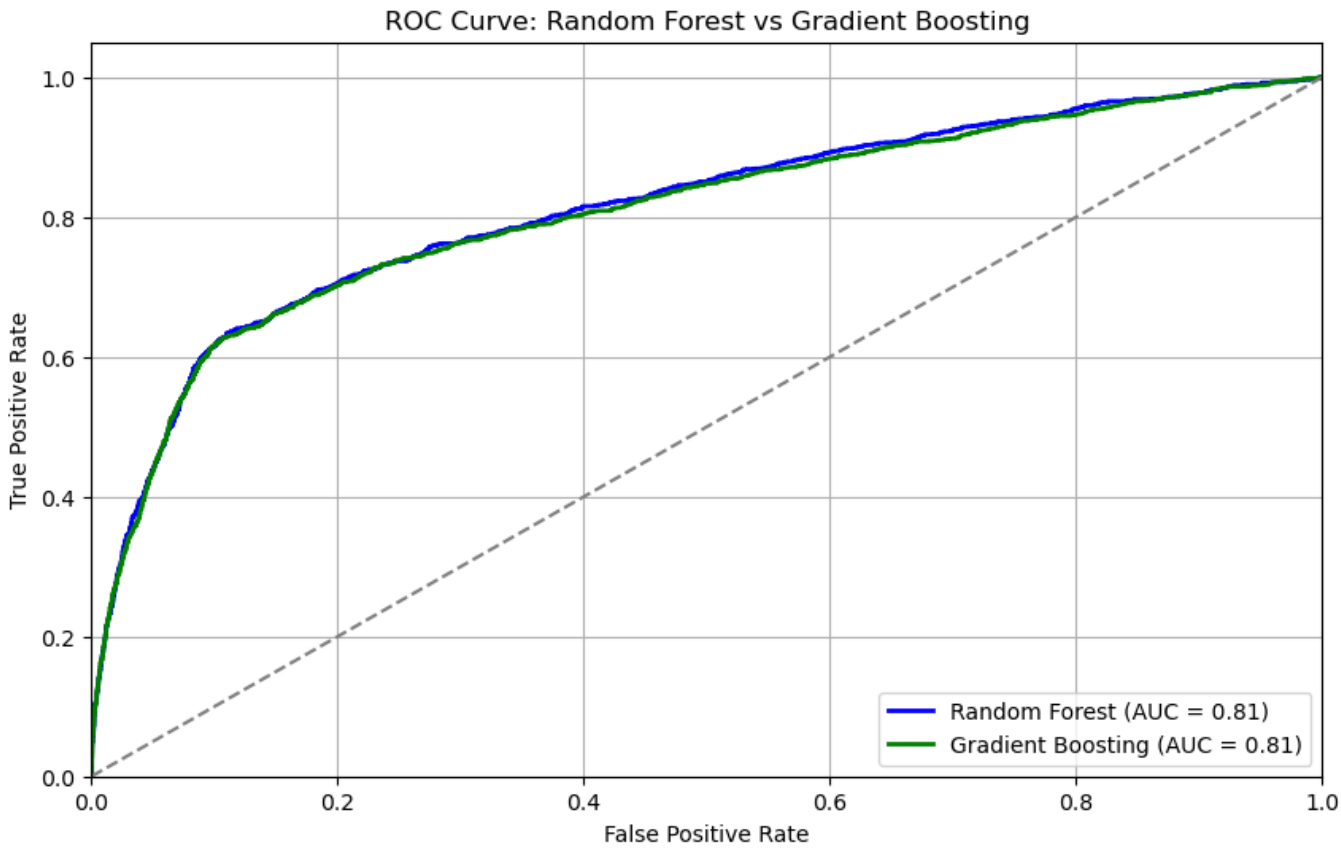
```
Best Gradient Boosting Performance:
[[10801   160]
 [ 1068   324]]
              precision    recall  f1-score   support

           0       0.91      0.99      0.95     10961
           1       0.67      0.23      0.35      1392

    accuracy                           0.90     12353
   macro avg       0.79      0.61      0.65     12353
weighted avg       0.88      0.90      0.88     12353
```

Observations:

The Gradient Boosting model achieved 11,125 correct predictions and 1,228 incorrect predictions, performing very well at identifying non-subscribers (class 0) but struggling with subscribers (class 1). The classification report shows an overall accuracy of 90%, with a precision of 90%, meaning the model is highly accurate when predicting subscribers. However, the recall for class 1 is low at 23%, indicating it misses many actual subscribers, resulting in a low F1-score of 35% for this class. These results suggest that while the model performs strongly overall, the class imbalance is affecting its ability to detect subscribers effectively.

Evaluating using AUC-ROC curve

The model is also evaluated using AUC (Area Under The Curve) ROC (Receiver Operating Characteristics) curve which is an evaluation metric for binary classification problems. An ROC curve is a graph showing the performance of a classifier. ROC is a probability curve plotted with True Positive Rate (also called Recall or Sensitivity) on the y-axis against False Positive Rate (also called as Precision) on the x-axis. The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve. The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.



7. Github Repo link