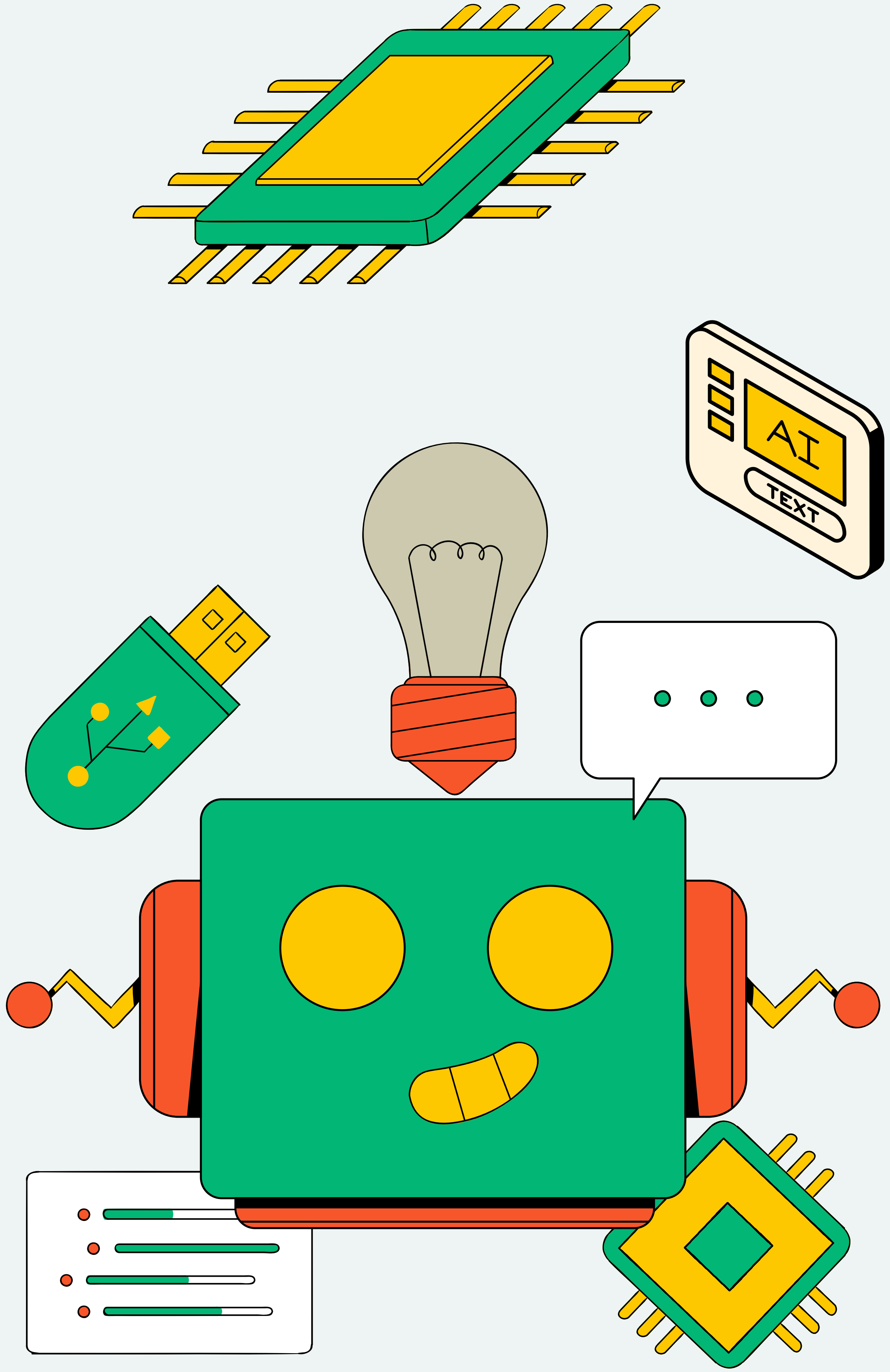


BANK MARKETING DATA ANALYSIS

PRESENTED BY:

VITALY SUKHININ

CHACK PU PATRICK TONG

KEXIN ZHU

OLUWATOSIN THOMAS

AGENDA

- Objective of the project
- Methodology
- Data Exploration
- Data preparation
- Machine Learning Implementation
- Compare and Conculde

INTRODUCTION

- This Project aims at predictive analytics in financial marketing.
- Analyzing the data of a Portuguese bank's marketing campaigns
- Forecast client engagement with term deposit subscriptions
- To determine the effectiveness of the campaign's success to other shareholders

OBJECTIVE OF THE PROJECT

- Analyze direct marketing campaigns
- Develop a predictive model
- Data cleaning
- Data preprocessing
- Machine learning modeling
- Evaluation

METHODOLOGY

DATA GATHERING & DATA ELPORATION

DATA VISUALIZATION

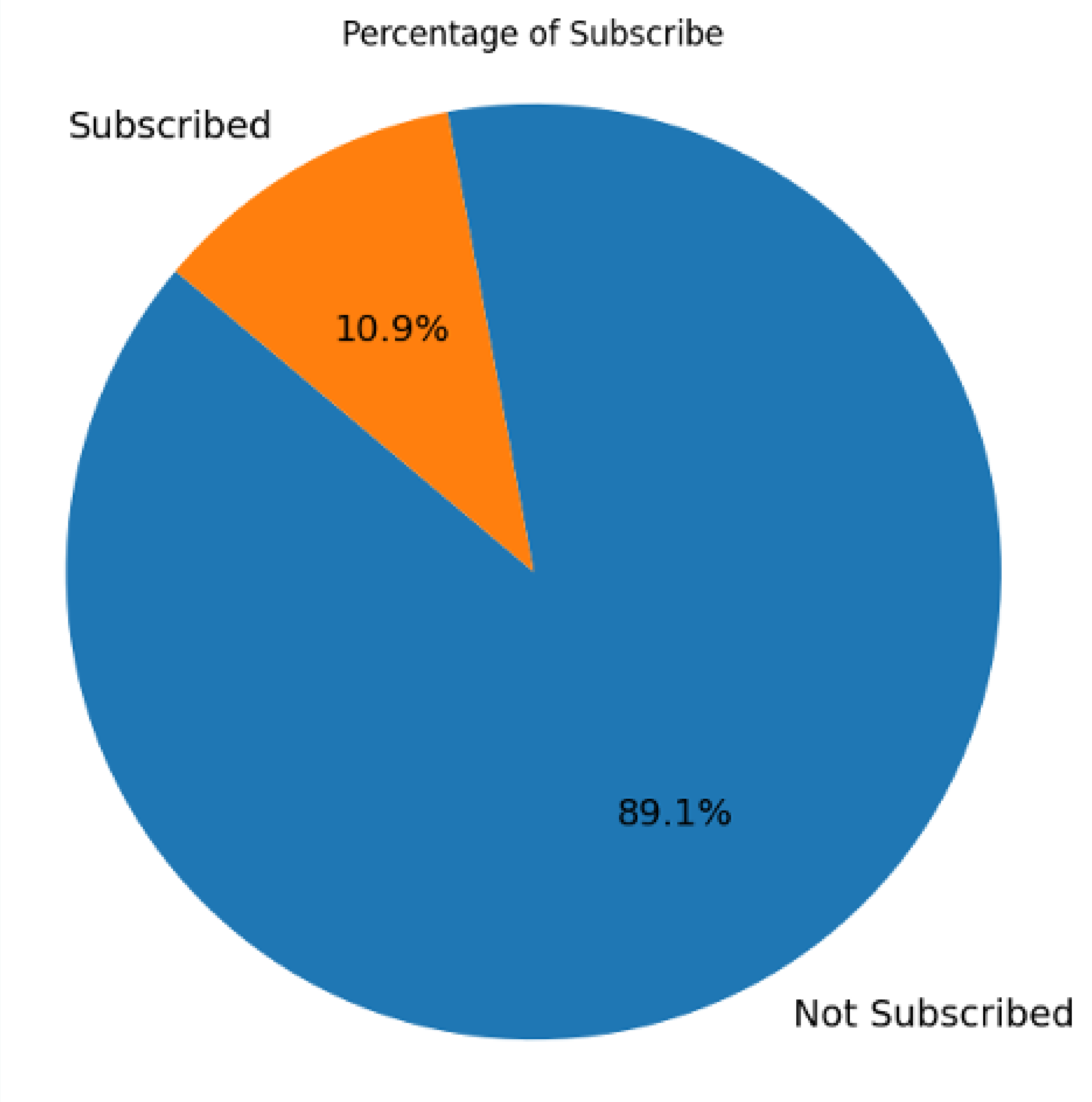
DATA CLEANING

DATA ENCODING

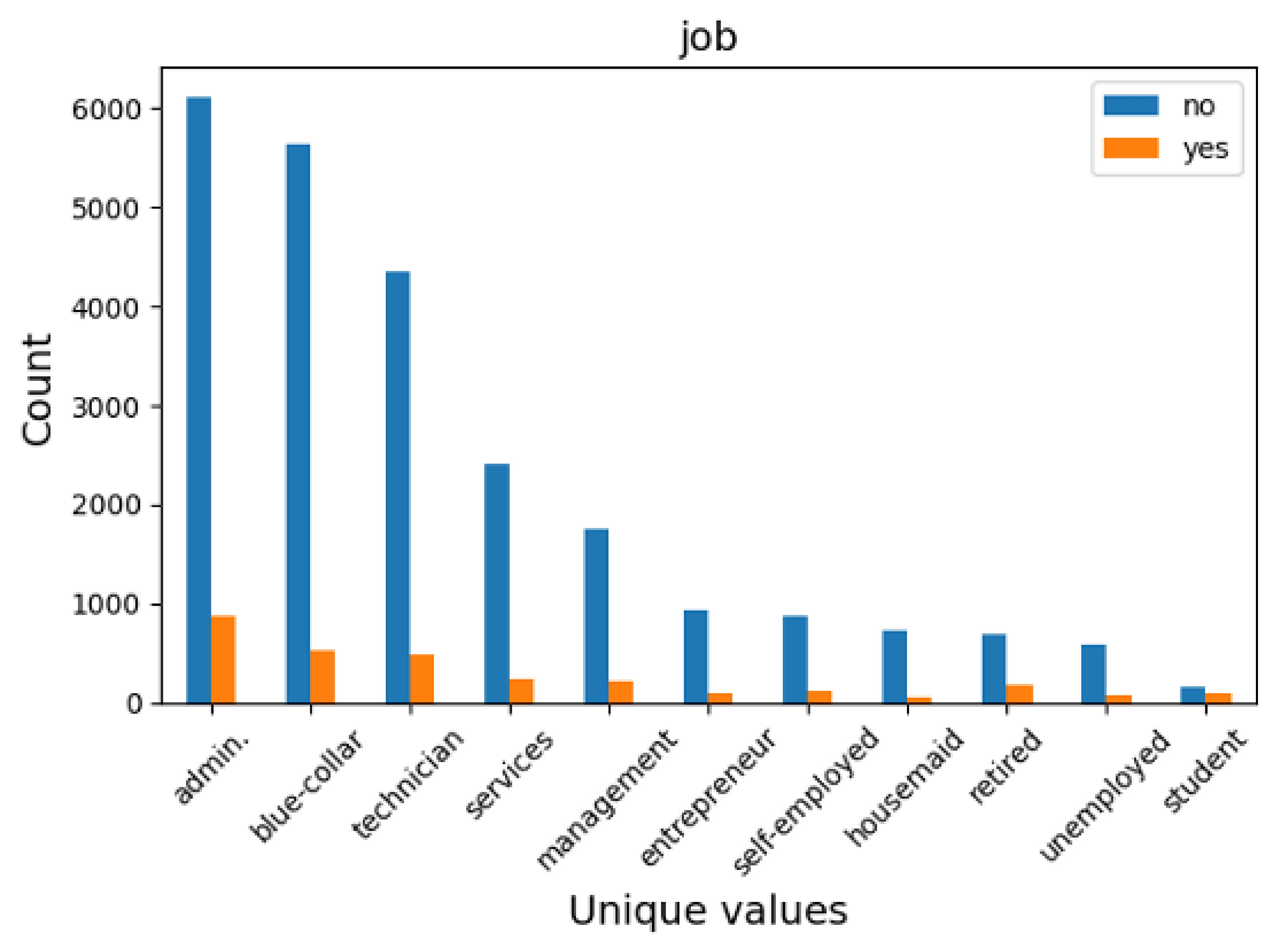
IMPLEMENT MACHINE LEARNING

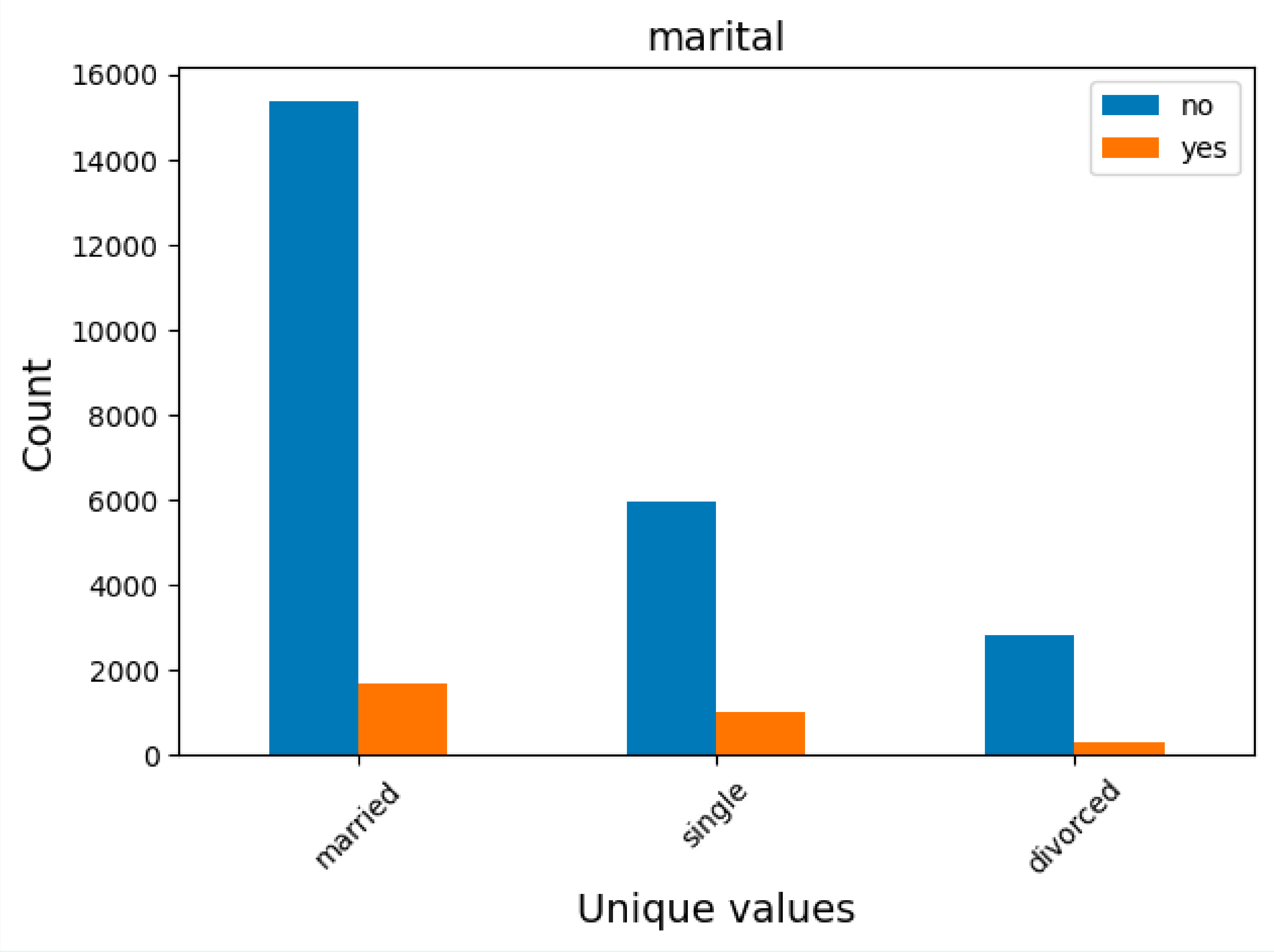
COMPARE AND CONCLUDE

DATA EXPLORATION

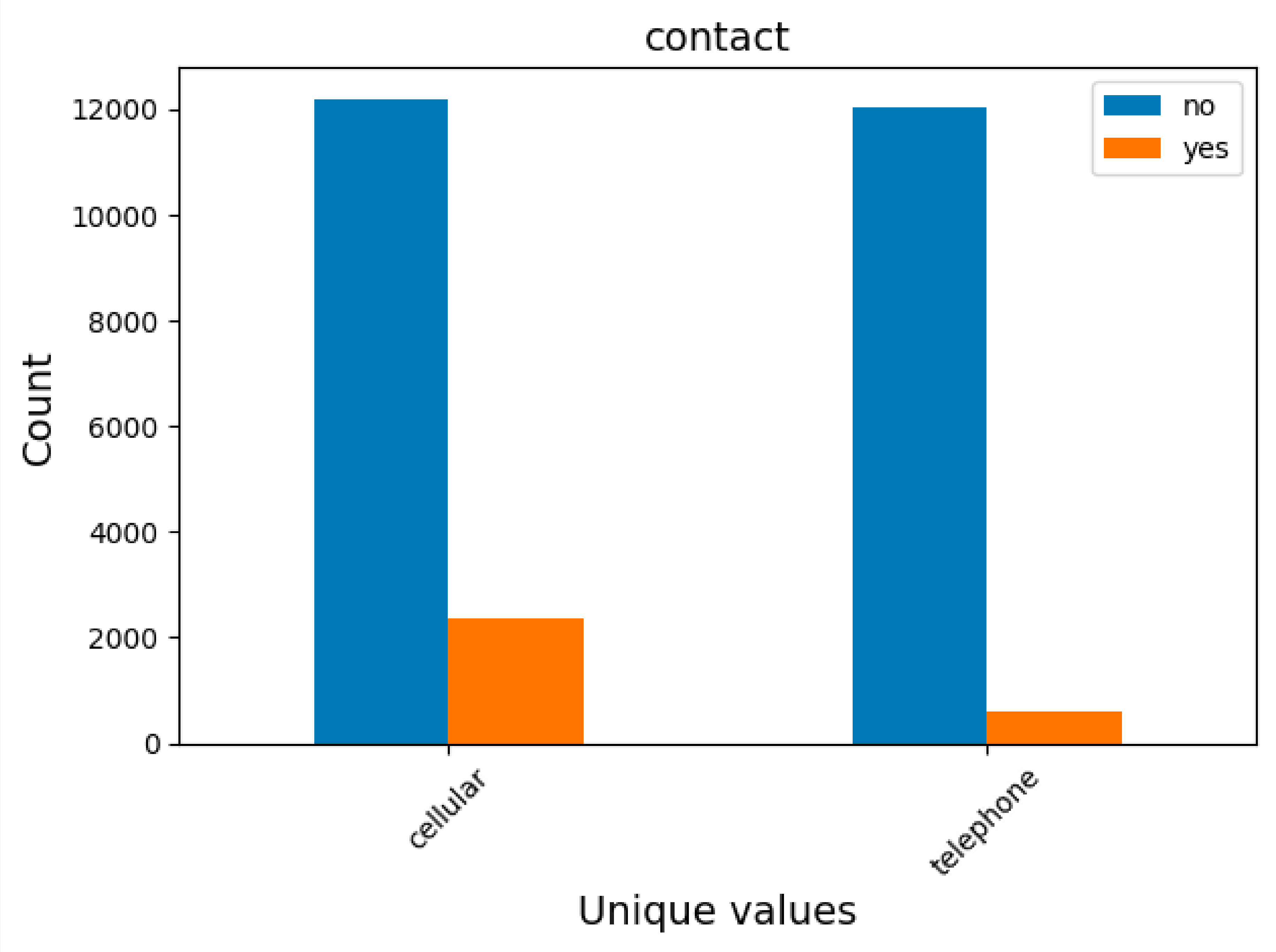


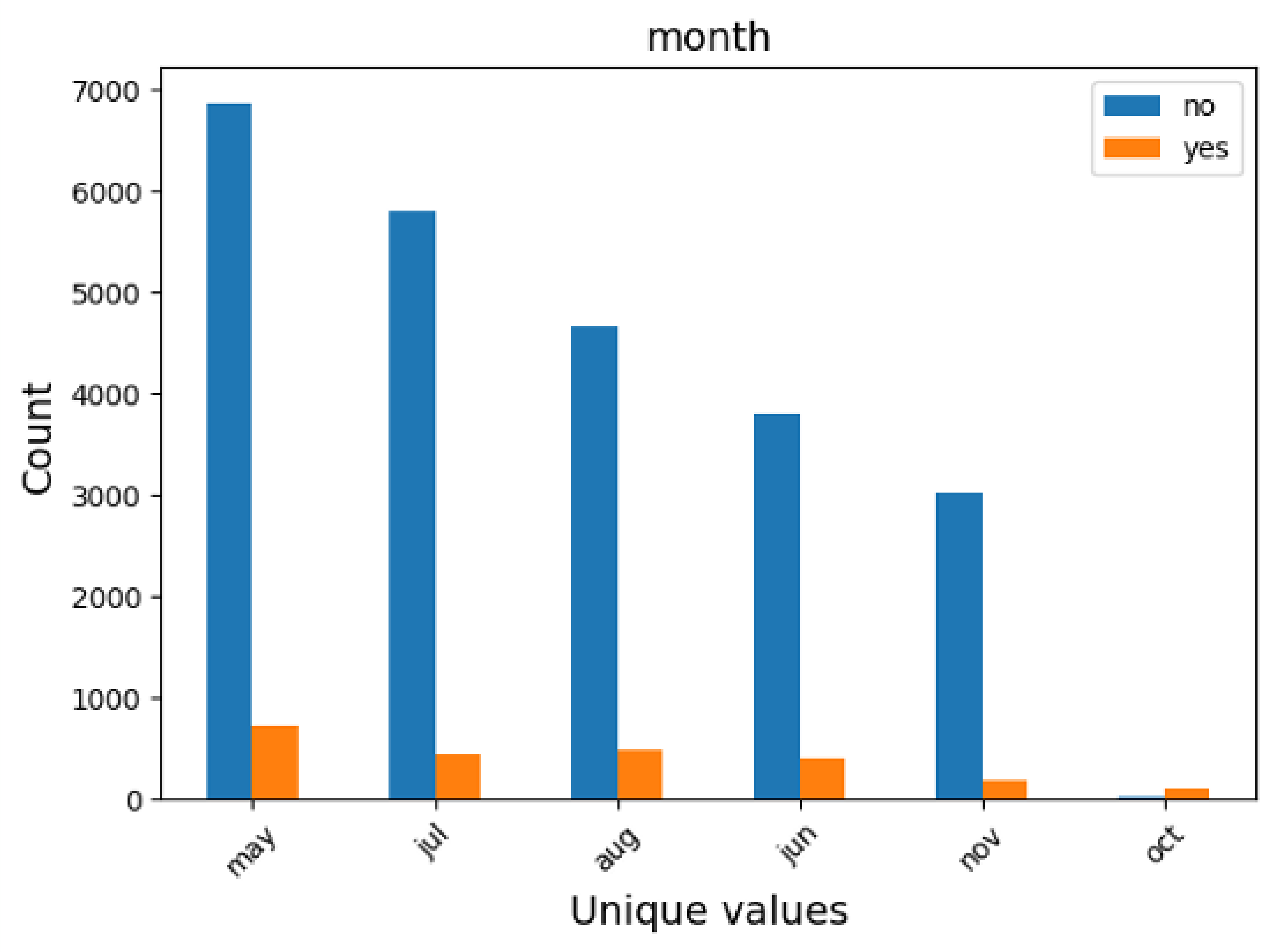
DATA EXPLORATION



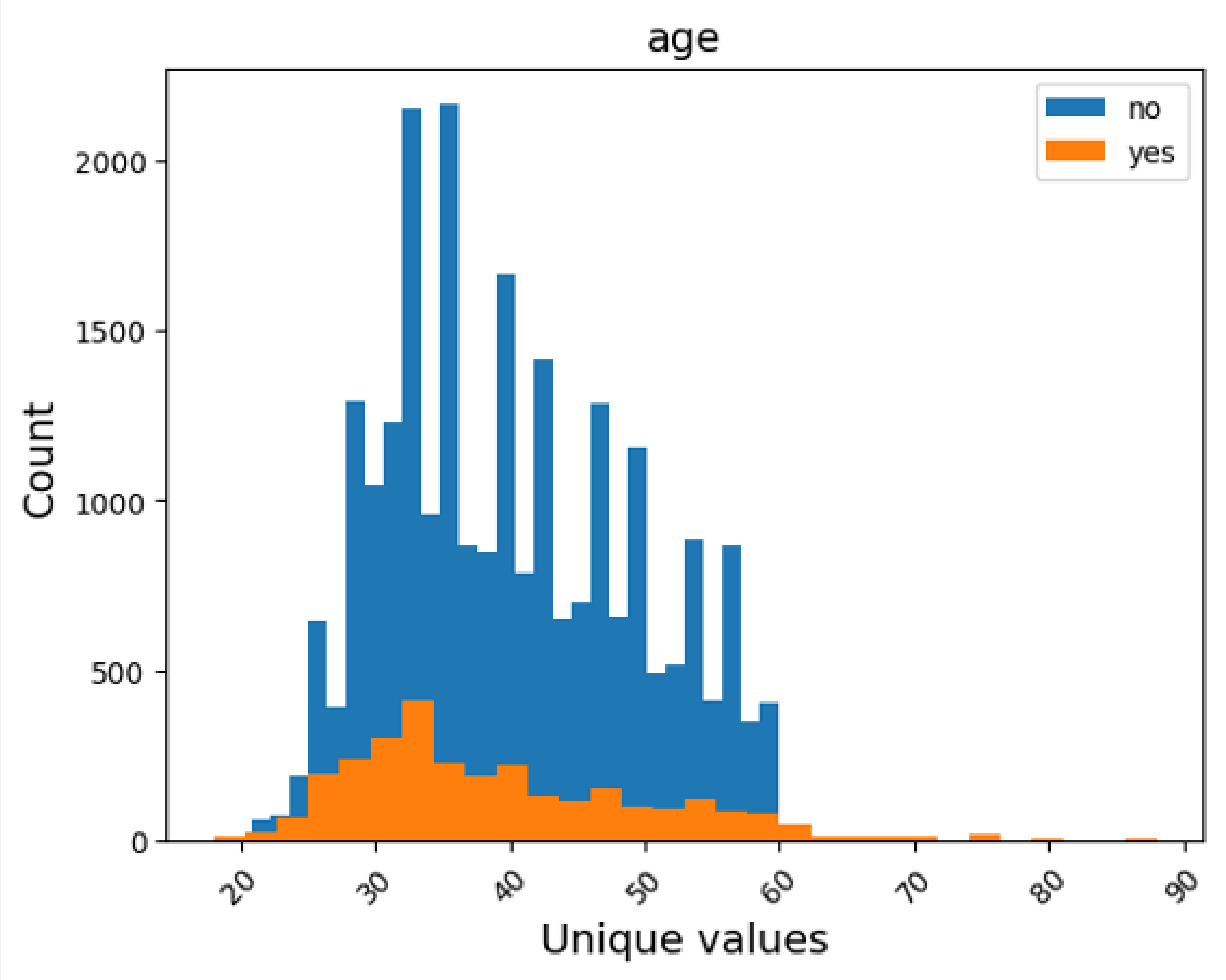


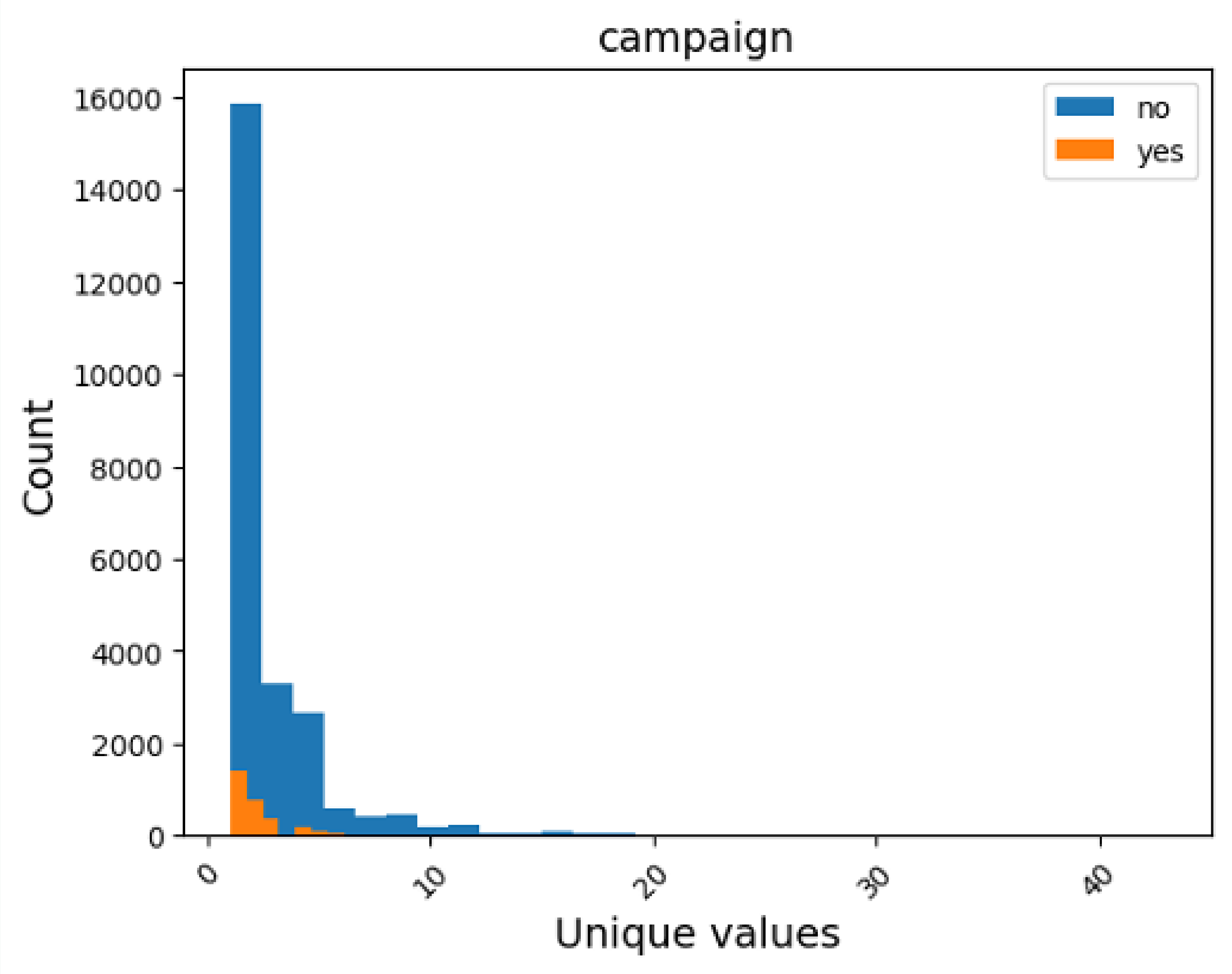
DATA EXPLORATION



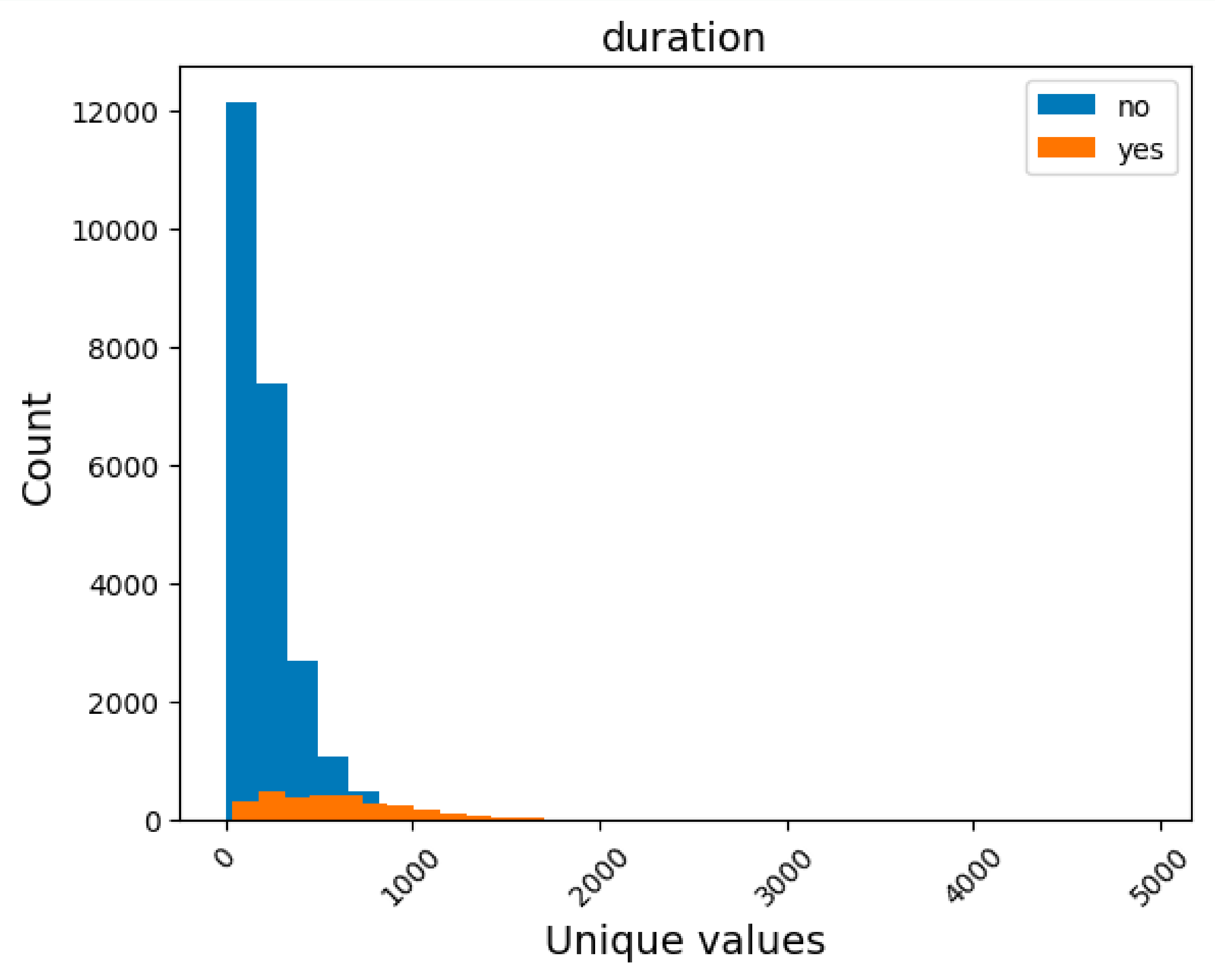


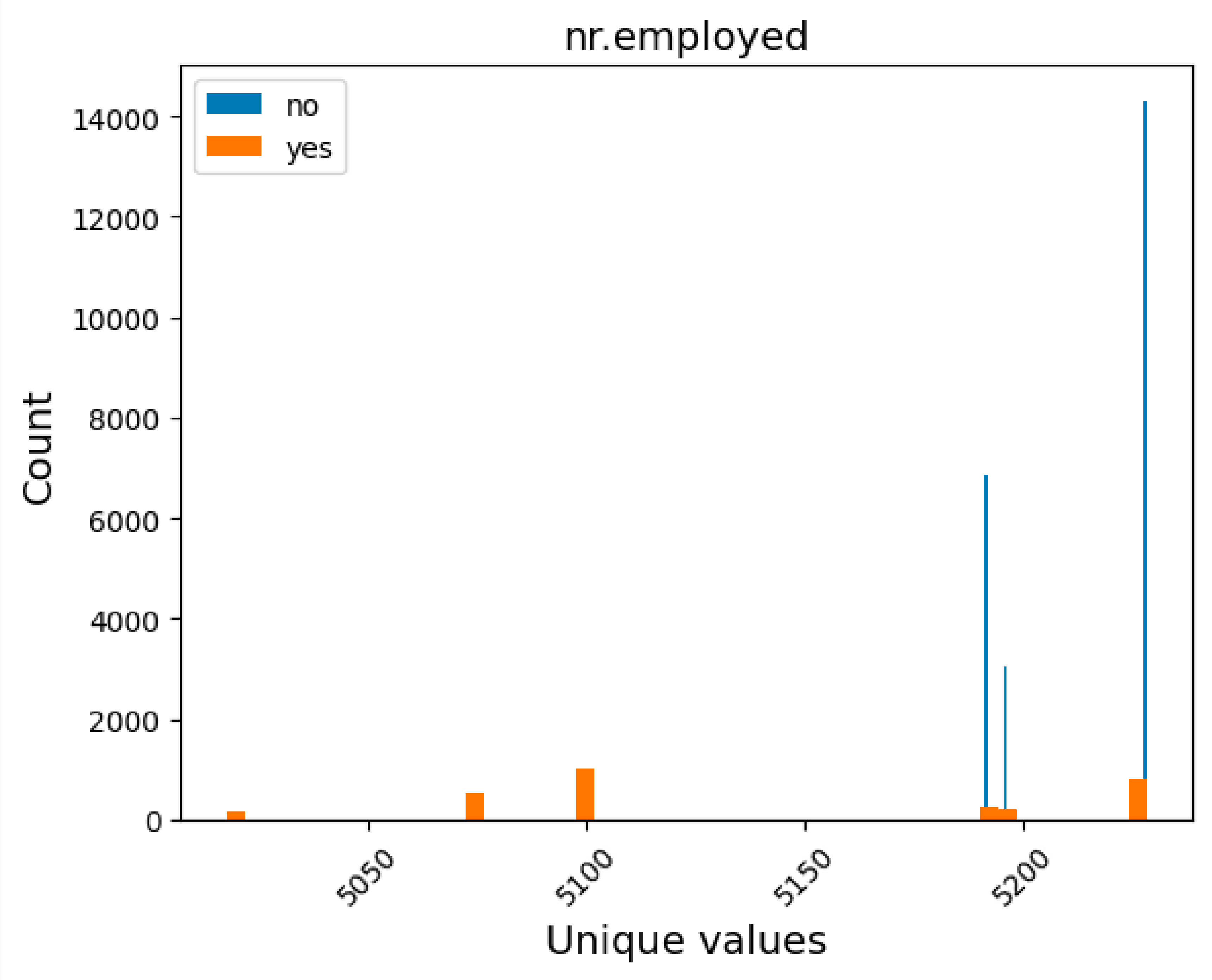
DATA EXPLORATION



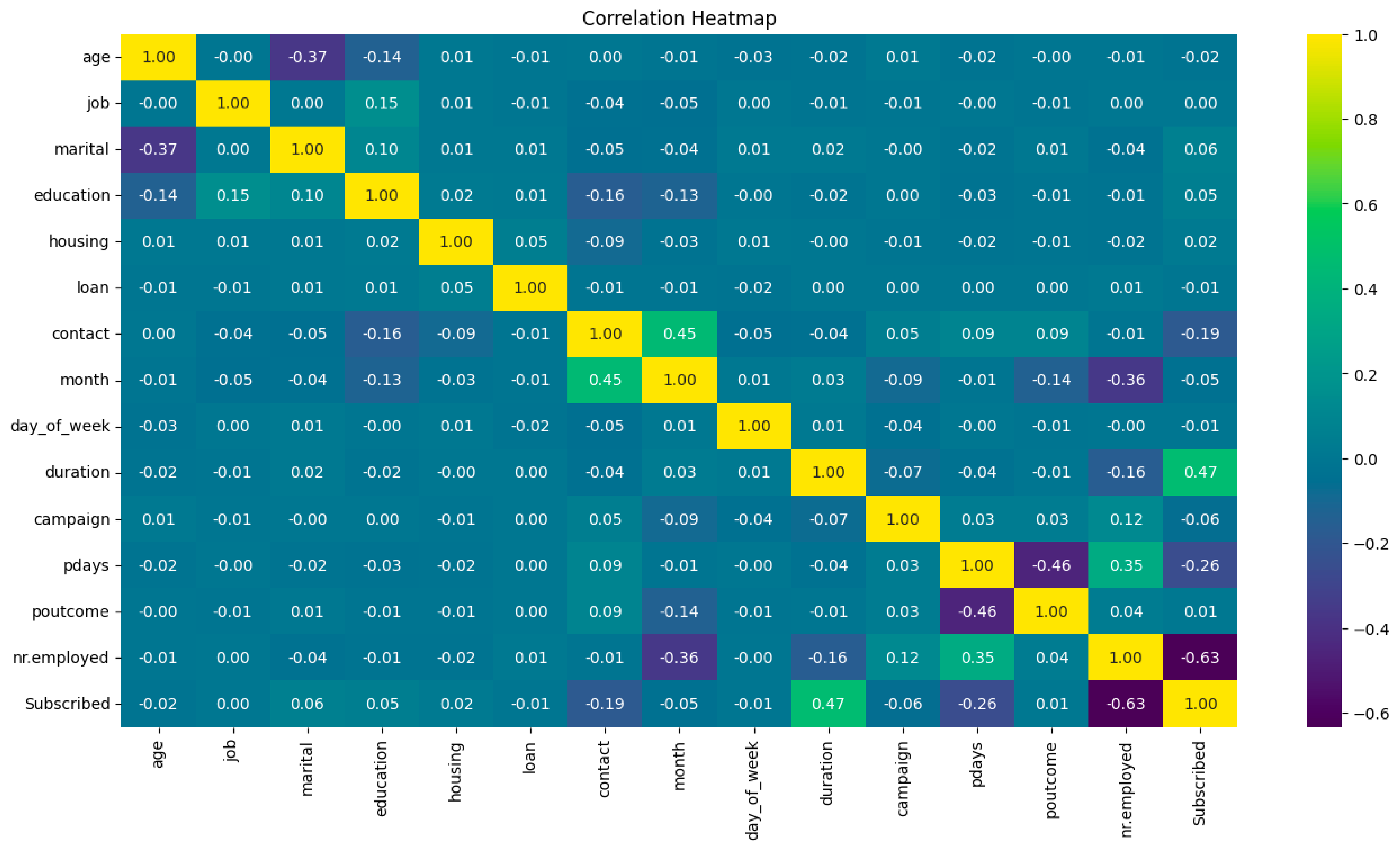


DATA EXPLORATION





DATA EXPLORATION



DATA CLEANING: 'UNKNOWN' VALUE

In [3]:

```python
raw_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29271 entries, 0 to 29270
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   age          29271 non-null  int64
 1   job          29271 non-null  object
 2   marital      29271 non-null  object
 3   education    29271 non-null  object
 4   housing      29271 non-null  object
 5   loan         29271 non-null  object
 6   contact      29271 non-null  object
 7   month        29271 non-null  object
 8   day_of_week  29271 non-null  object
 9   duration     29271 non-null  int64
 10  campaign     29271 non-null  int64
 11  pdays        29271 non-null  int64
 12  poutcome     29271 non-null  object
 13  nr.employed  29271 non-null  float64
 14  Subscribed   29271 non-null  object
dtypes: float64(1), int64(4), object(10)
memory usage: 3.3+ MB
```

In [4]:

```python
# covert unknown Data to np.Nan
raw_data = raw_data.replace('unknown',np.nan)
raw_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29271 entries, 0 to 29270
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   age          29271 non-null  int64
 1   job          29011 non-null  object
 2   marital      29220 non-null  object
 3   education    28044 non-null  object
 4   housing      28558 non-null  object
 5   loan         28558 non-null  object
 6   contact      29271 non-null  object
 7   month        29271 non-null  object
 8   day_of_week  29271 non-null  object
 9   duration     29271 non-null  int64
 10  campaign     29271 non-null  int64
 11  pdays        29271 non-null  int64
 12  poutcome     29271 non-null  object
 13  nr.employed  29271 non-null  float64
 14  Subscribed   29271 non-null  object
dtypes: float64(1), int64(4), object(10)
memory usage: 3.3+ MB
```

DATA CLEANING: 'UNKNOWN' VALUE

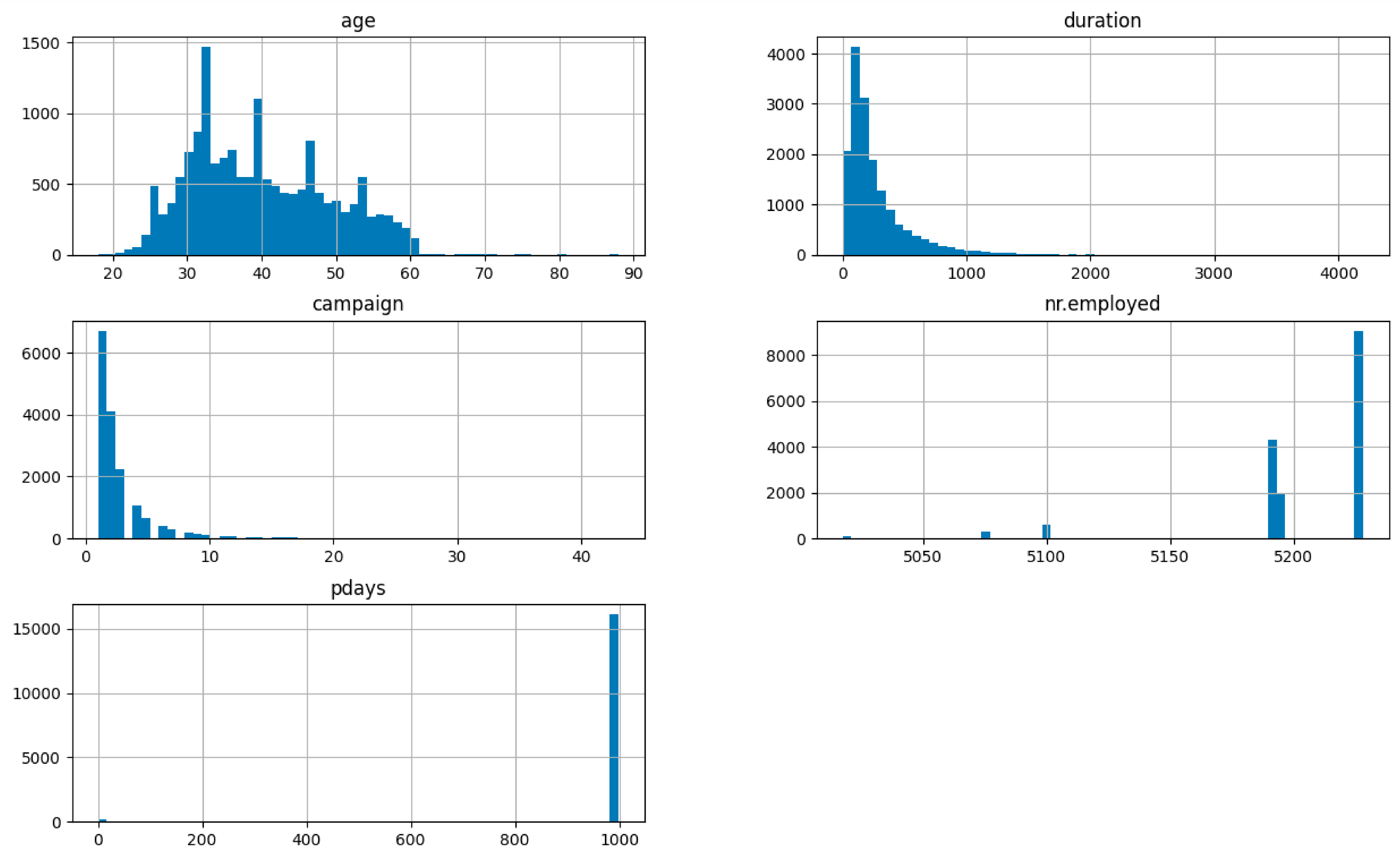
In [5]:
```
# Check Empty Data
raw_data["job"].unique()
raw_data.isna().sum()
```

Out[5]:
```
age               0
job             260
marital          51
education      1227
housing         713
loan            713
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
poutcome          0
nr.employed       0
Subscribed        0
dtype: int64
```

```
#Check no of unknown
raw_data = raw_data.dropna(subset=["job","marital","education","housing","loan"])
raw_data.isna().sum()
```

Out[6]:
```
age            0
job            0
marital        0
education      0
housing        0
loan           0
contact        0
month          0
day_of_week    0
duration       0
campaign       0
pdays          0
poutcome       0
nr.employed    0
Subscribed     0
dtype: int64
```

DATA CLEANING: OUTLINERS

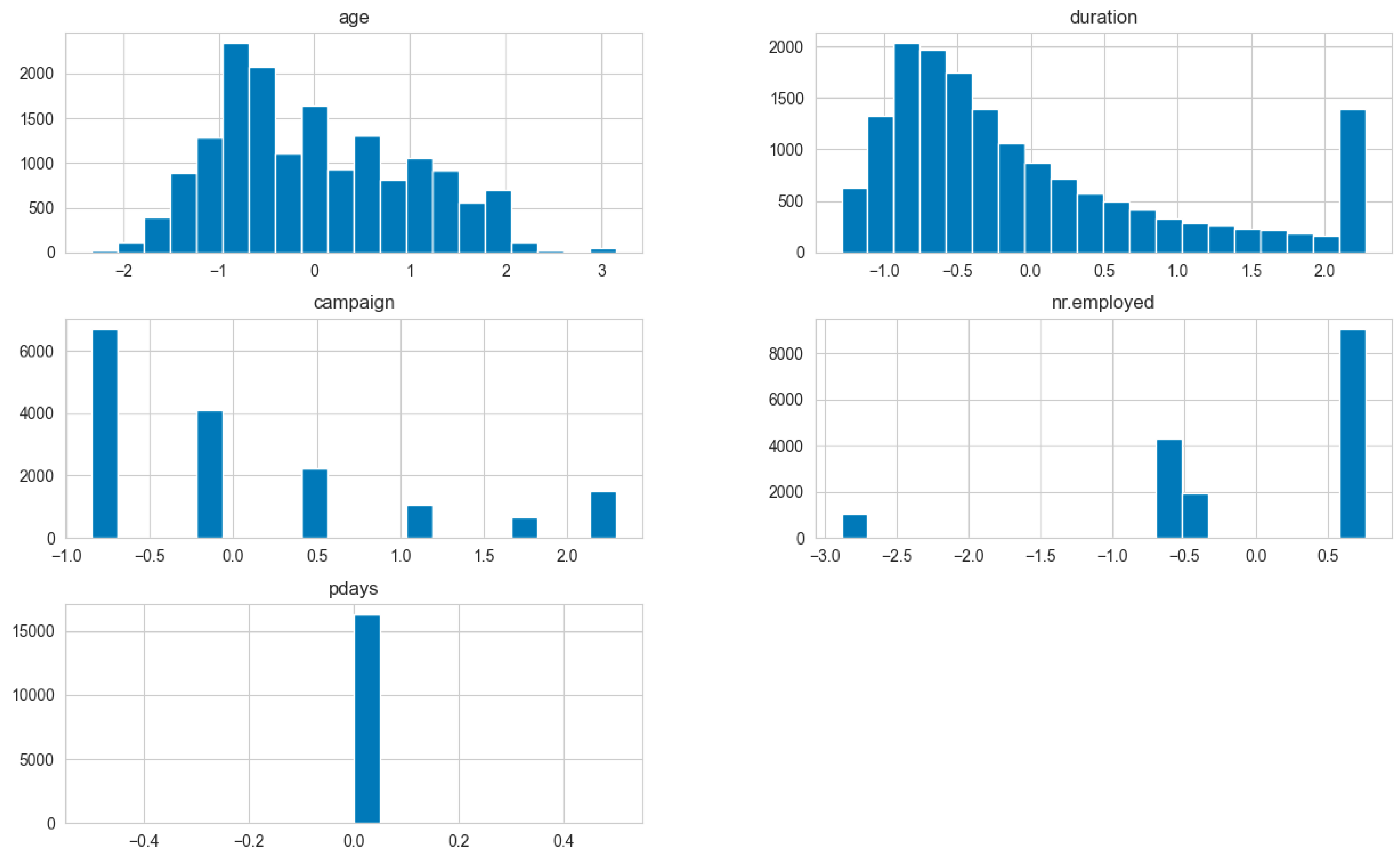


DATA PREPARATION: STANDARDIZATION

```
#Standarization
from sklearn.preprocessing import StandardScaler
std_scaler = StandardScaler()
std_scaler.fit(int_data)
int_data_scaled = std_scaler.transform(int_data
df_int_std = pd.DataFrame(int_data_scaled, colu
df_int_std.head()
```

✓ 0.0s

	age	duration	campaign	nr.employed	p
0	-0.836502	1.776431	-0.846475	-2.881411	
1	-0.410540	-0.456802	-0.846475	0.762704	
2	-1.049483	0.047646	2.290485	0.762704	
3	0.015422	-0.582914	-0.846475	0.762704	
4	0.121913	-0.057448	-0.846475	-0.506352	



DATA PREPARATION: ENCODING

Encoding strategy

Identify Data Type and separate

- Ordinary Data
- Nominal Data

Encoding with following

- ONE HOT ENCODING
- ORDINAL ENCODING

```
#Converting Ordinary data to array index using OrdinaryEncoder
from sklearn.preprocessing import OrdinalEncoder
ordinal_Encoder = OrdinalEncoder(categories=[['illiterate', 'basic.4y','basic.6y', 'basic.9y','high.school', '
       'professional.course']])
cat_ordinal_endcoded = ordinal_Encoder.fit_transform(cat_ordinal)
df_ordinal_endcoded = pd.DataFrame(cat_ordinal_endcoded,columns= cat_ordinal.columns)
df_ordinal_endcoded.astype(float)
df_ordinal_endcoded.head()
```

	education
0	4.0
1	2.0
2	3.0
3	5.0
4	4.0

In [22]:

```
#Converting Nominal data to array index using one hot encoder
from sklearn.preprocessing import OneHotEncoder
oneHotEncoder = OneHotEncoder()
oneHotEncoder.fit(cat_nominal)
cat_nominal_1hot = oneHotEncoder.transform(cat_nominal)

df_nominal_1hot = pd.DataFrame(cat_nominal_1hot.toarray(), columns=oneHotEncoder.get_feature_names_out())
df_nominal_1hot
```

Out[22]:

	month_apr	month_aug	month_dec	month_jul	month_jun	month_mar	month_may	month_nov	month_oct	mo
0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	

DATA PREPARATION: ENCODING

After separately processing numerical and categorical features, the data was recombined into a single dataframe.

In [24]:

```
# Concat all the standardize data and encoded data
x_train = pd.concat([df_int_std,df_ordinal_endcoded,df_nominal_1hot],axis = 1)
```

In [25]:

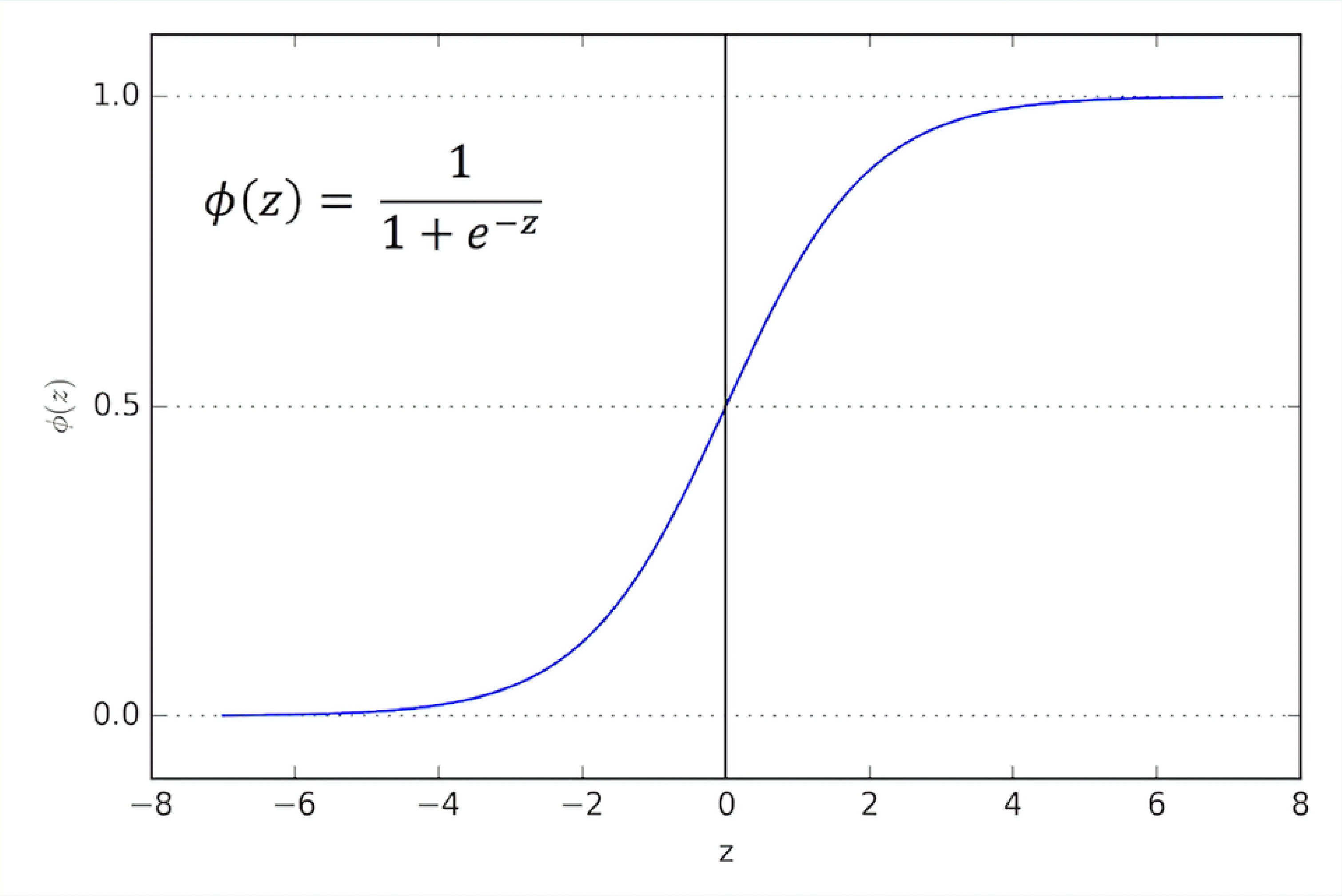
```
x_train.head()
```

Out[25]:

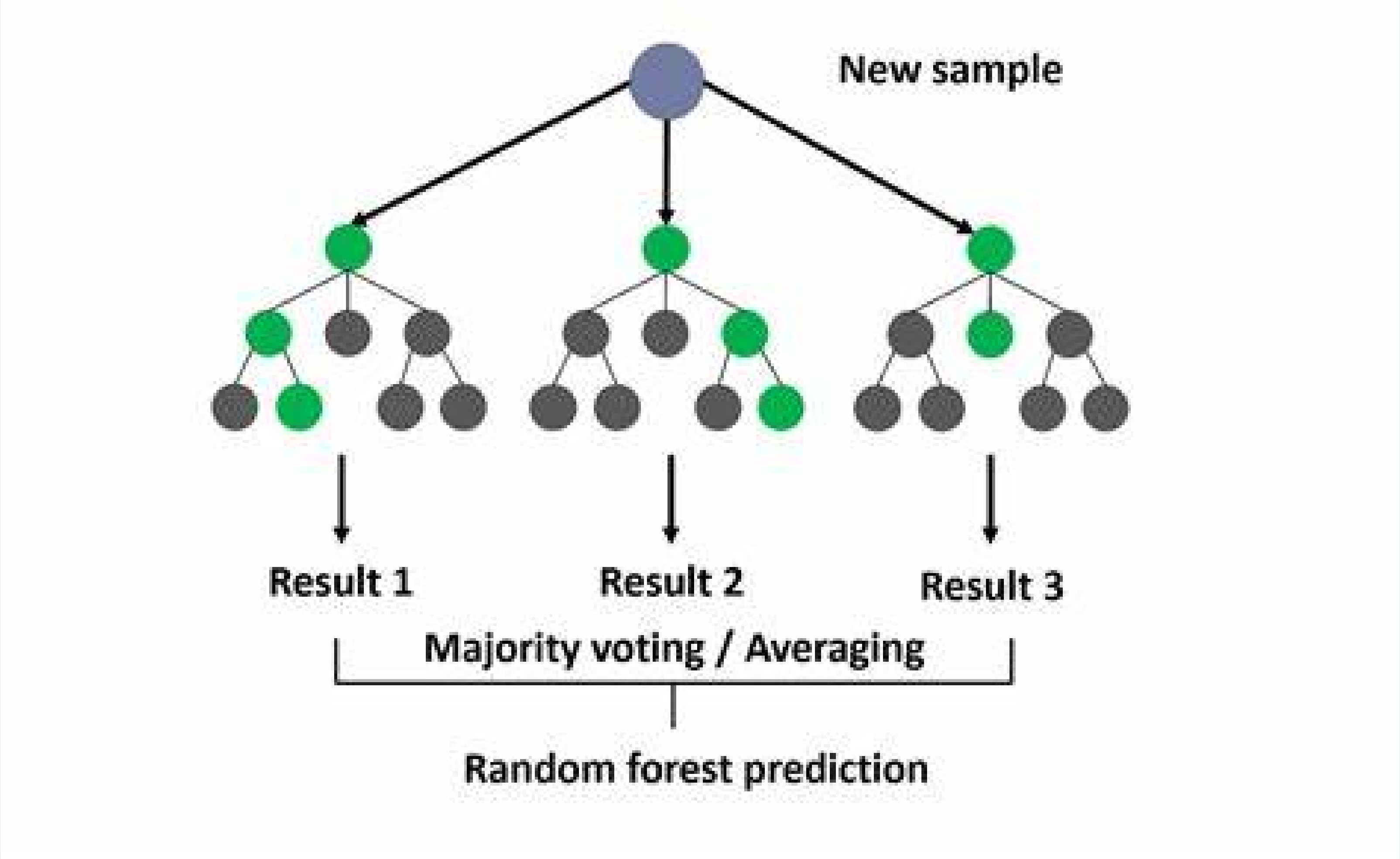
	age	duration	campaign	nr.employed	pdays	education	month_apr	month_aug	month_dec	month_jul
0	-0.521317	1.900333	-0.563989	0.627573	0.103854	4.0	0.0	0.0	0.0	1.0
1	1.171907	-0.055572	-0.563989	-0.398550	0.103854	2.0	0.0	0.0	0.0	0.0
2	0.748601	0.858402	-0.235533	0.627573	0.103854	3.0	0.0	0.0	0.0	0.0
3	0.960254	0.054105	0.092922	-5.197265	-9.650522	5.0	0.0	0.0	0.0	0.0
4	-1.897060	-0.373635	-0.235533	-3.573723	0.103854	4.0	0.0	0.0	0.0	0.0

LEARNING METHODS

Logistic Regression



Random Forest Classification



MACHINE LEARNING IMPLEMENTATION

Prepare Testset for prediction on trained machine learning Model

In [28]:
```
test_data_ordinal = ordinal_Encoder.transform(x_test[ordinal_column])
test_data_nominal = oneHotEncoder.transform(x_test[nominal_column])
test_data_int = std_scaler.transform(x_test[int_column])
test_data_result = YoneHotEncoder.transform(pd.DataFrame(y_test))
```

In [29]:
```
df_test_nominal_1hot = pd.DataFrame(test_data_nominal.toarray(), columns=oneHotEncoder.get_feature_names_out())
df_test_ordinal_endcoded = pd.DataFrame(test_data_ordinal,columns= cat_ordinal.columns)
df_test_data_int = pd.DataFrame(test_data_int, columns= std_scaler.get_feature_names_out())
```

In [30]:
```
#Concat all Transformed Data
x_test  = pd.concat([df_test_data_int,df_test_ordinal_endcoded,df_test_nominal_1hot],axis = 1)
```

MACHINE LEARNING IMPLEMENTATION

In [30]:
```
#Concat all Transformed Data
x_test  = pd.concat([df_test_data_int,df_test_ordinal_endcoded,df_test_nominal_1hot],axis = 1)
x_test.head()
```

Out[30]:

	age	duration	campaign	nr.employed	pdays	education	month_apr	month_aug	month_dec	month_jul	...	mari
0	1.171907	0.090664	1.406743	0.627573	0.103854	3.0	0.0	0.0	0.0	1.0	...	
1	-0.415490	0.072384	-0.563989	0.627573	0.103854	4.0	0.0	0.0	0.0	1.0	...	
2	-0.415490	2.565707	-0.563989	-2.940348	0.103854	5.0	1.0	0.0	0.0	0.0	...	
3	-0.203837	-0.761160	1.078288	0.627573	0.103854	2.0	0.0	0.0	0.0	0.0	...	
4	0.748601	-0.614924	-0.563989	-2.940348	0.103854	6.0	1.0	0.0	0.0	0.0	...	

Comparing training and test set

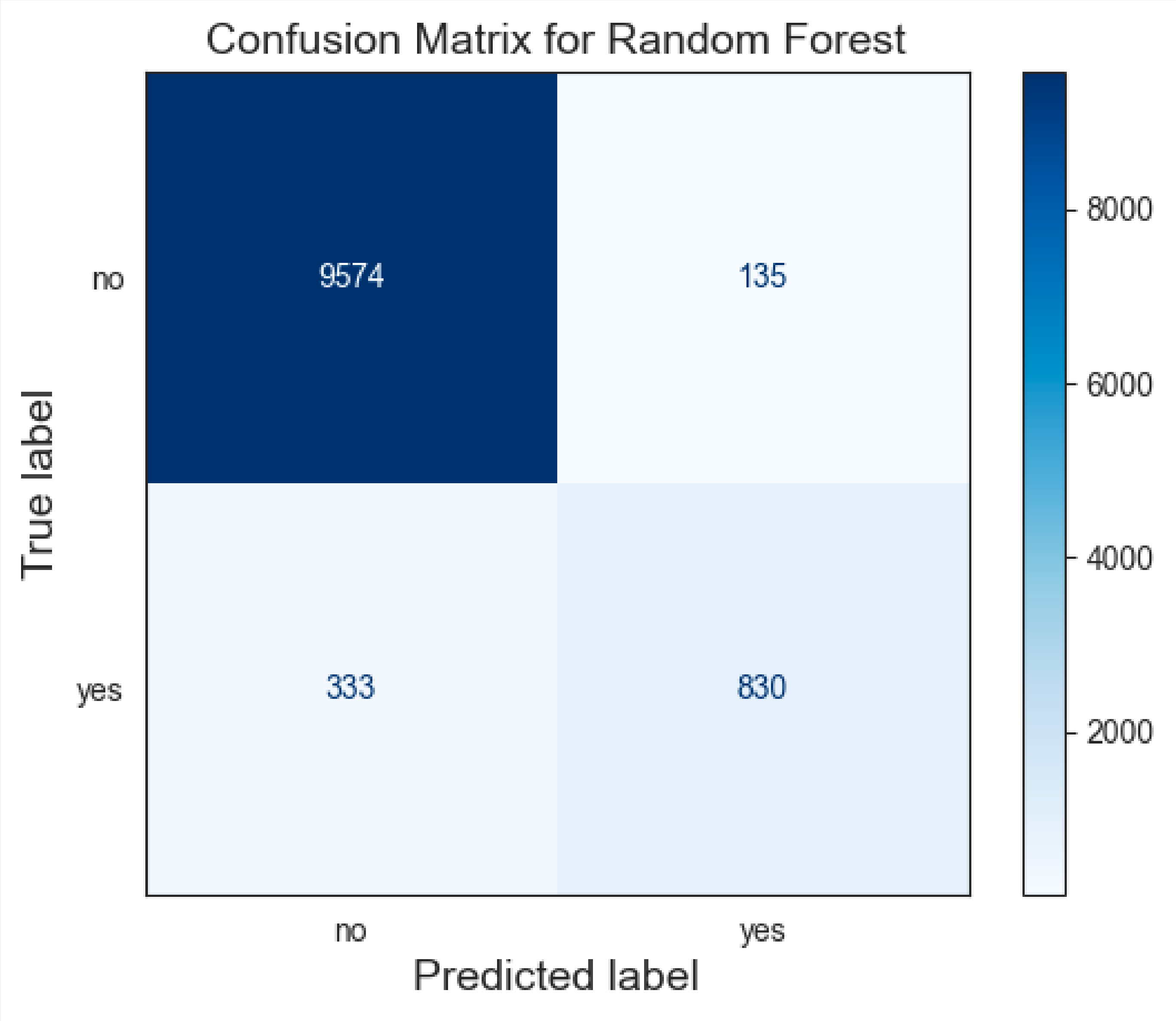
In [31]:
```
#Compare to train Data
x_train.head()
```

Out[31]:

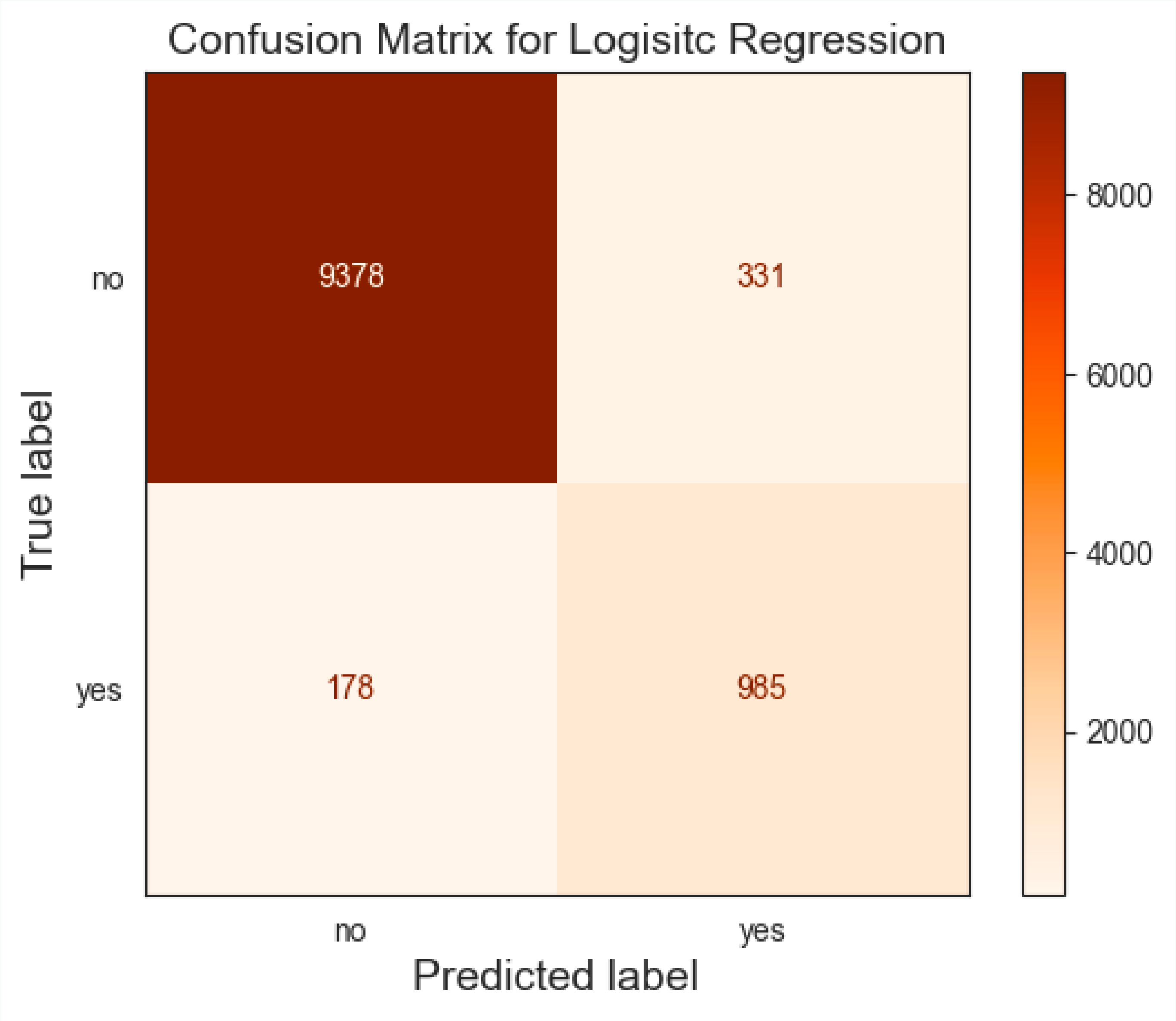
	age	duration	campaign	nr.employed	pdays	education	month_apr	month_aug	month_dec	month_jul	...	mar
0	-0.521317	1.900333	-0.563989	0.627573	0.103854	4.0	0.0	0.0	0.0	1.0	...	
1	1.171907	-0.055572	-0.563989	-0.398550	0.103854	2.0	0.0	0.0	0.0	0.0	...	
2	0.748601	0.858402	-0.235533	0.627573	0.103854	3.0	0.0	0.0	0.0	0.0	...	
3	0.960254	0.054105	0.092922	-5.197265	-9.650522	5.0	0.0	0.0	0.0	0.0	...	
4	-1.897060	-0.373635	-0.235533	-3.573723	0.103854	4.0	0.0	0.0	0.0	0.0	...	

MACHINE LEARNING IMPLEMENTATION: PREDICTIONS

Random Forest Classification



Logistic Regression



MACHINE LEARNING IMPLEMENTATION: EVALUATION

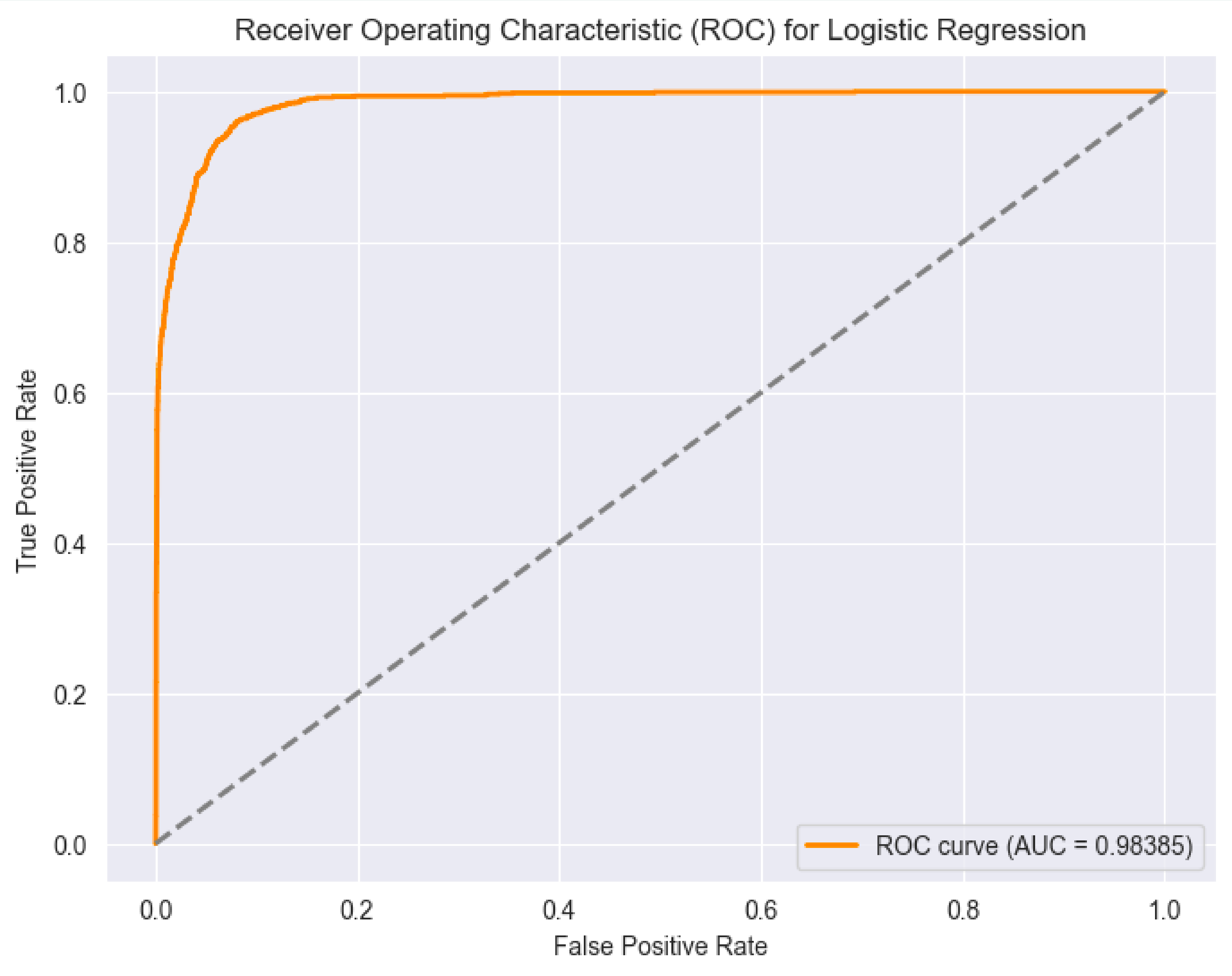
Random Forest Classification

	precision	recall	f1-score	support
no	0.97	0.99	0.98	9709
yes	0.86	0.71	0.78	1163
accuracy			0.96	10872
macro avg	0.91	0.85	0.88	10872
weighted avg	0.96	0.96	0.96	10872

Logistic Regression

	precision	recall	f1-score	support
no	0.98	0.97	0.97	9709
yes	0.75	0.85	0.79	1163
accuracy			0.95	10872
macro avg	0.86	0.91	0.88	10872
weighted avg	0.96	0.95	0.95	10872

MACHINE LEARNING IMPLEMENTATION: EVALUATION



MACHINE LEARNING IMPLEMENTATION: EVALUATION

- Both Model has a high accuracy with around 96%
- Random Forest has a higher f1 score on ’no’
- Logistic Regression has a higher f1 score on ’yes’
- Slightly higher AUC on the Logistic Regression
- Logistics Regression is a better model approach

CONCLUSION

- the imbalance in subscription outcomes, significant features influencing subscription
- The logistic Model has a higher F1 score and AUC which Suggest to be a higher accuracy model
- The project allow compare the effectiveness of the campaign with other campaign in comping future.

THANK YOU