Data Glacier

Your Deep Learning Partner

Bank Marketing Campaign

29.12.2023

Data Glacier

Your Deep Learning Partner

Group Name	Tentacool
Name	Murat Kıran
Email	murattkiran@gmail.com
Country	Türkiye
Specialization	Data Science

Problem Statement

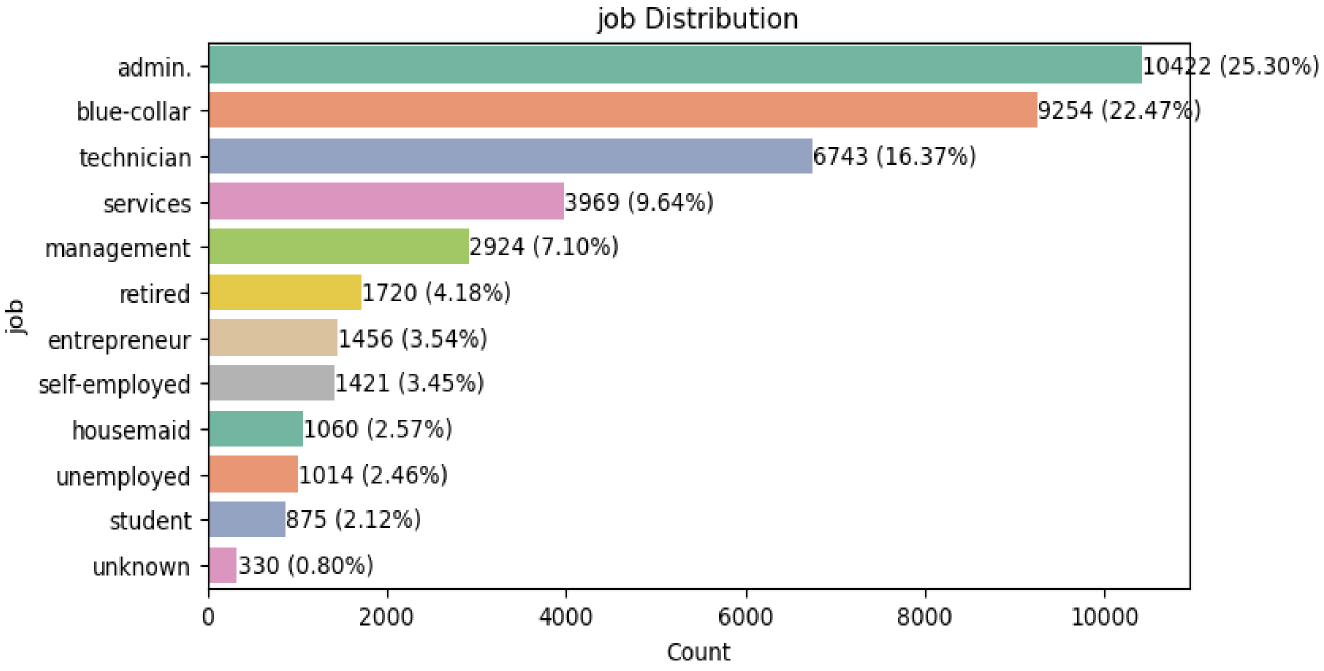
- ABC Bank is on the verge of launching a new term deposit product and aims to boost its success by developing a predictive model.
- The goal is to identify whether a customer will subscribe to the term deposit ('yes') or not ('no') based on past interactions.
- The challenge lies in optimizing marketing efforts and tailoring strategies to maximize customer engagement.
- The dataset, derived from Portuguese banking campaigns, contains various client details and campaign outcomes.
- The objective is to create a robust predictive model that provides insights into factors influencing subscription decisions, empowering ABC Bank to refine its marketing approach for the impending product launch.

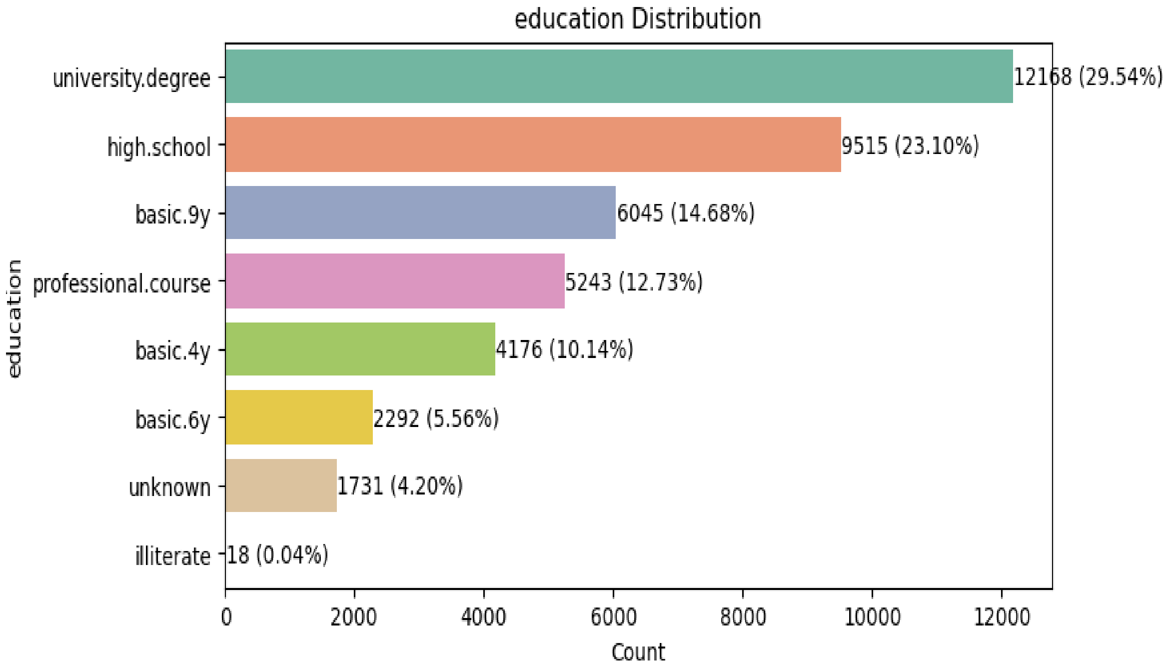
Dataset

- Total number of observations 41188.
- There is no missing value in this dataset.
- **Categorical Columns**: 'job', 'marital', 'education', 'default', 'housing', 'loan', 'contact', 'month', 'day_of_week', 'poutcome‘
- **Numerical Columns:** 'age', 'campaign', 'pdays', 'previous', 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed'
- The **duration** feature was removed from the dataset to avoid data leakage. This attribute highly affects the output target, and its value is known only after the call is performed, leading to unrealistic predictive models. The removal aligns with the intention to develop a realistic predictive model.

Job

Education

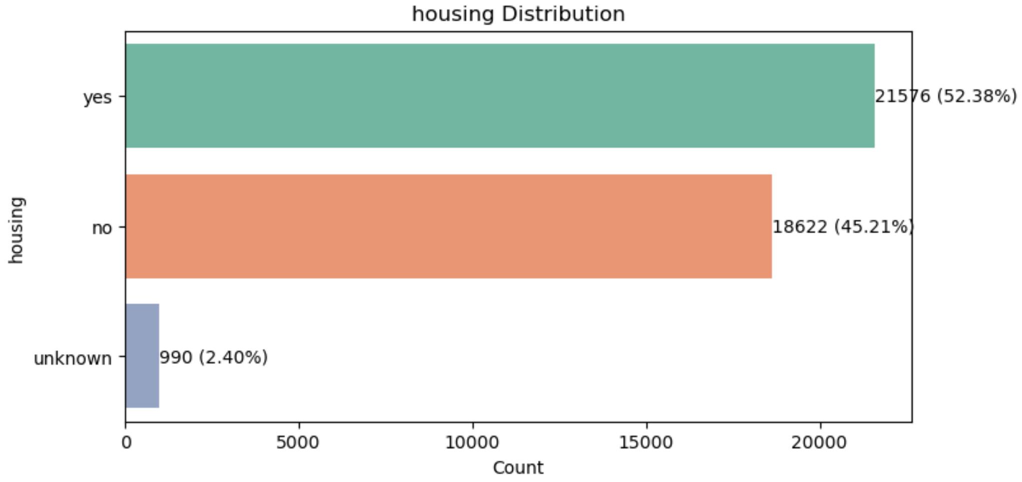




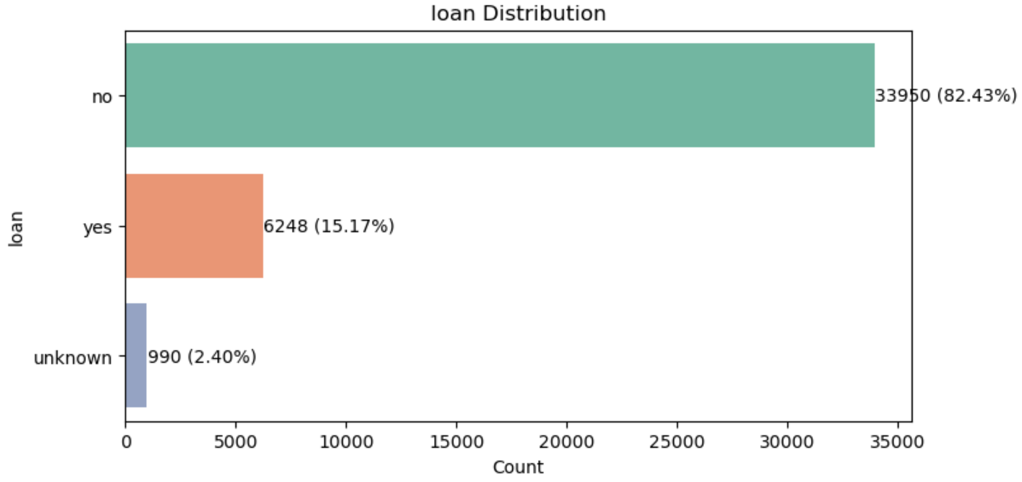
job	y_mean
admin.	0.129726
blue-collar	0.068943
entrepreneur	0.085165
housemaid	0.100000
management	0.112175
retired	0.252326
self-employed	0.104856
services	0.081381
student	0.314286
technician	0.108260
unemployed	0.142012
unknown	0.112121

education	y_mean
basic.4y	0.102490
basic.6y	0.082024
basic.9y	0.078246
high.school	0.108355
illiterate	0.222222
professional.course	0.113485
university.degree	0.137245
unknown	0.145003

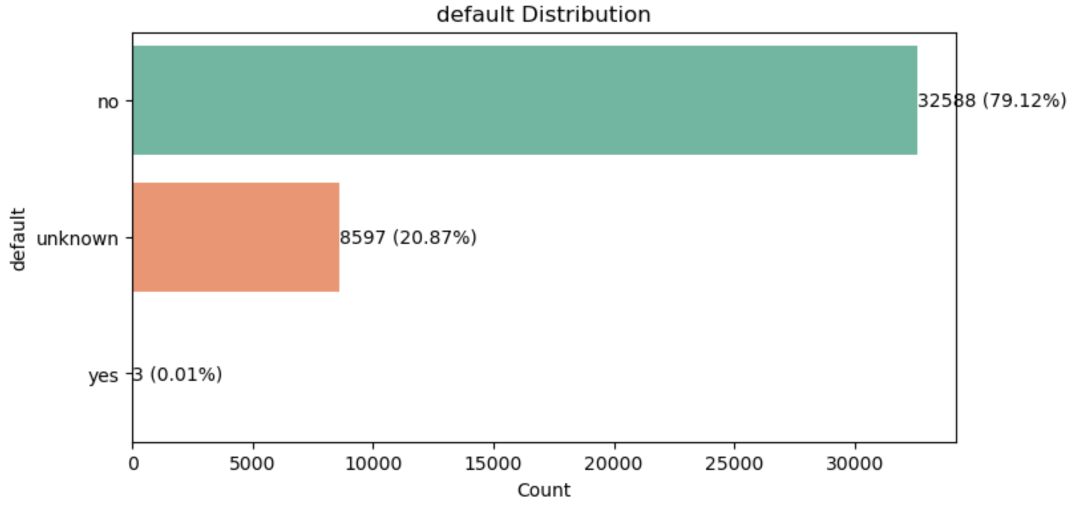
Default, housing and loan



	y_mean
housing	
no	0.108796
unknown	0.108081
yes	0.116194



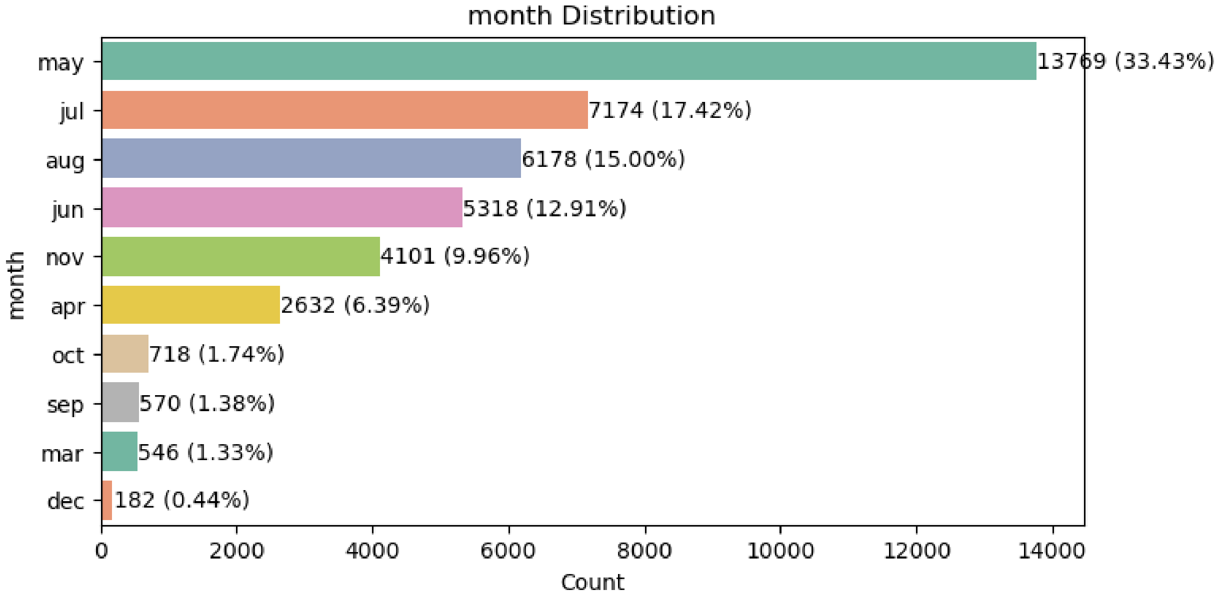
	y_mean
loan	
no	0.113402
unknown	0.108081
yes	0.109315

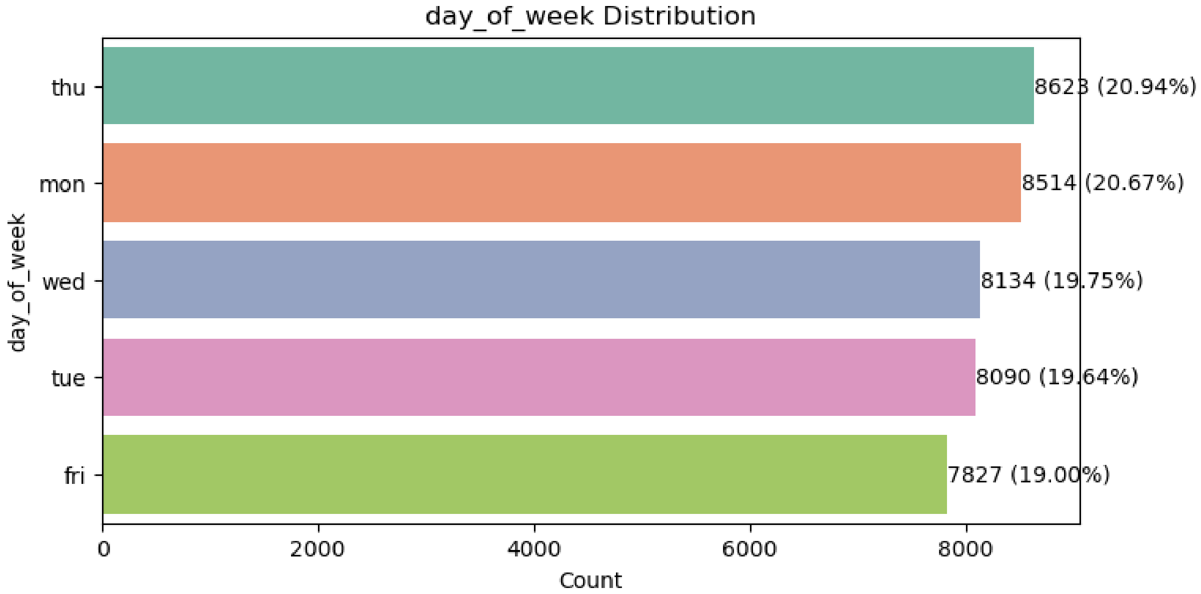


	y_mean
default	
no	0.12879
unknown	0.05153
yes	0.00000

Months

Days



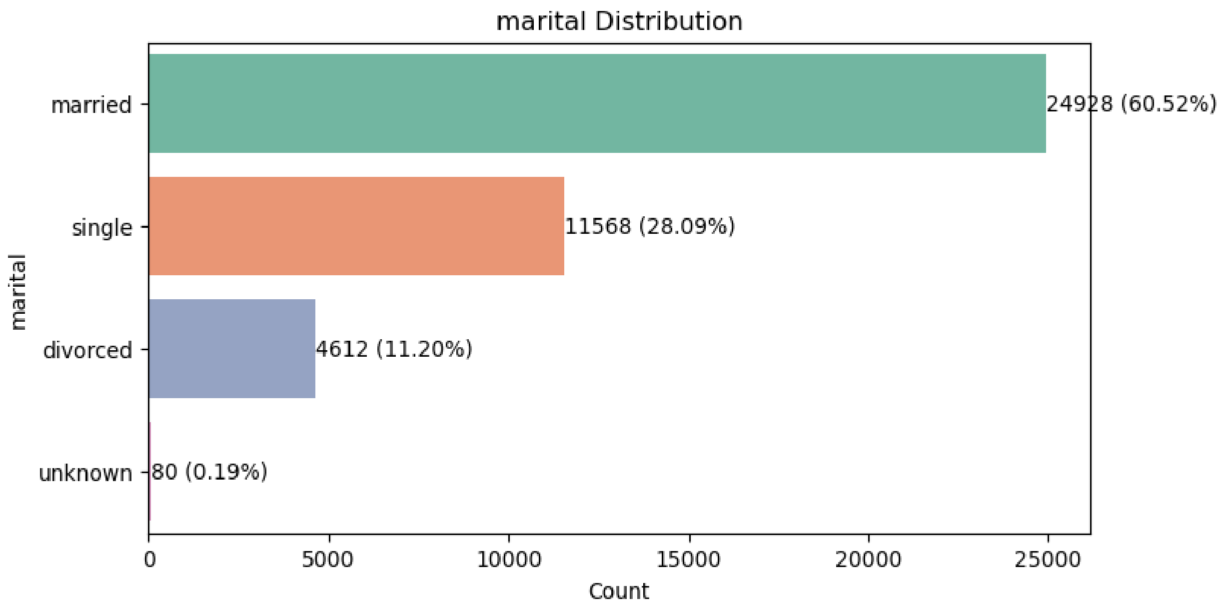


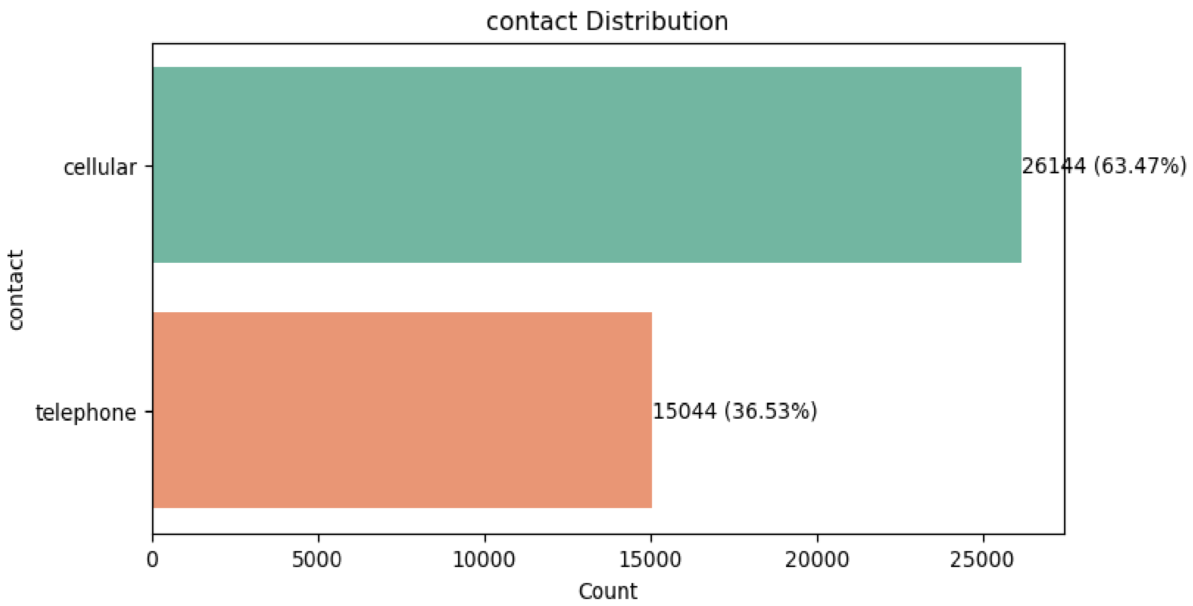
month	y_mean
apr	0.204787
aug	0.106021
dec	0.489011
jul	0.090466
jun	0.105115
mar	0.505495
may	0.064347
nov	0.101439
oct	0.438719
sep	0.449123

day_of_week	y_mean
fri	0.108087
mon	0.099483
thu	0.121188
tue	0.117800
wed	0.116671

Marital Status

Contact



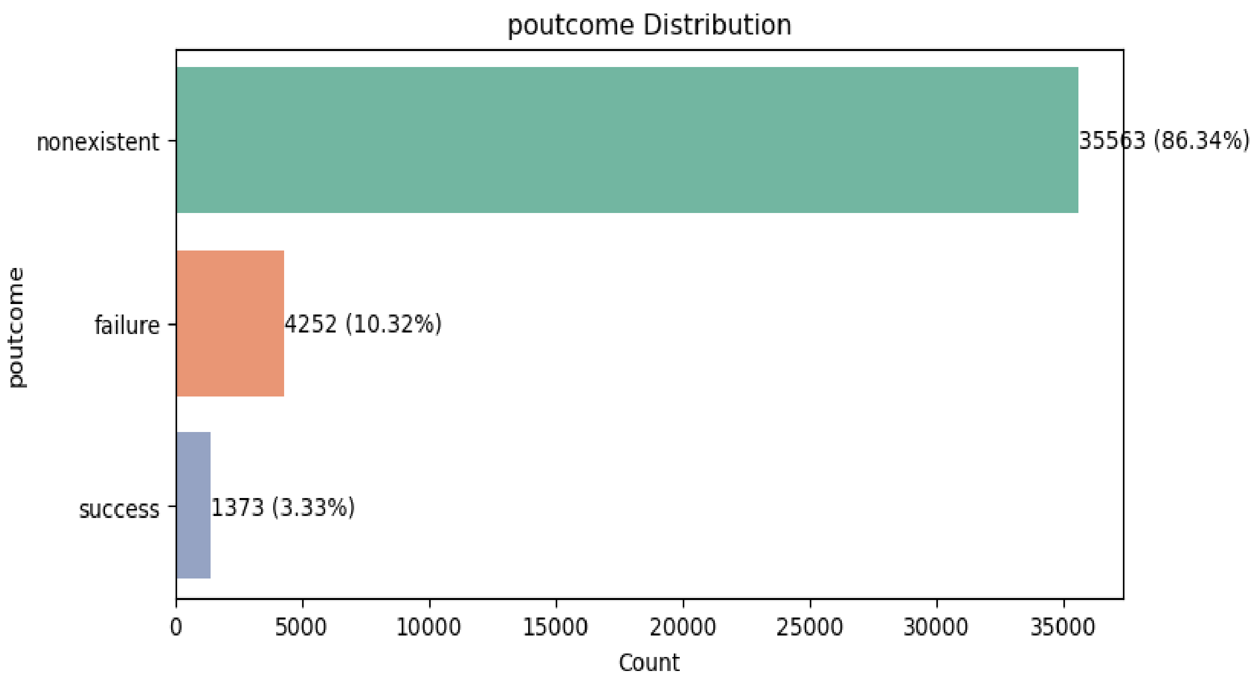


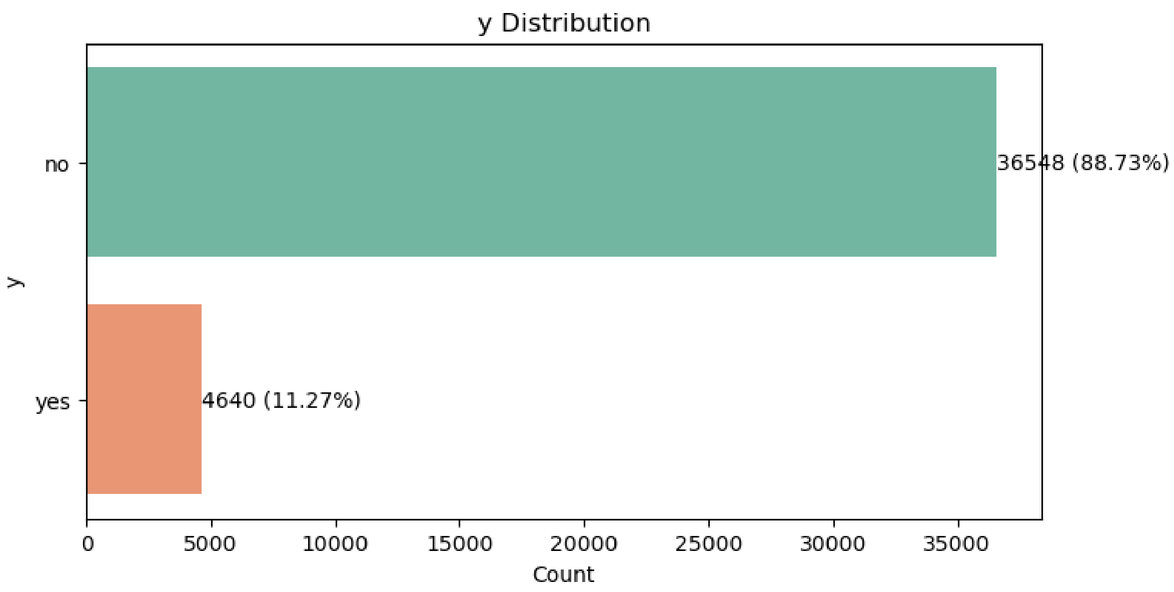
marital	y_mean
divorced	0.103209
married	0.101573
single	0.140041
unknown	0.150000

contact	y_mean
cellular	0.147376
telephone	0.052313

Poutcome

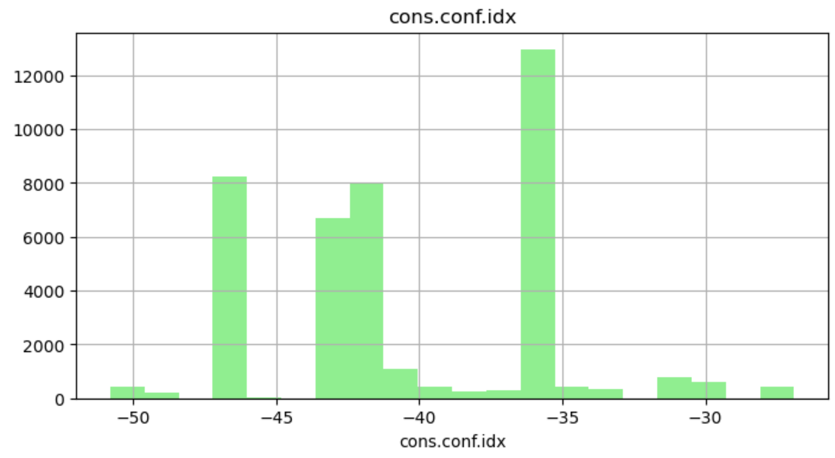
Target variable

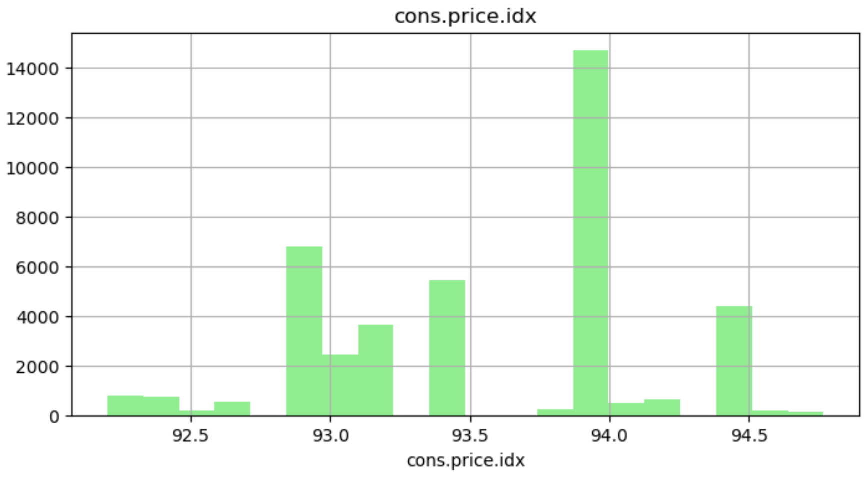


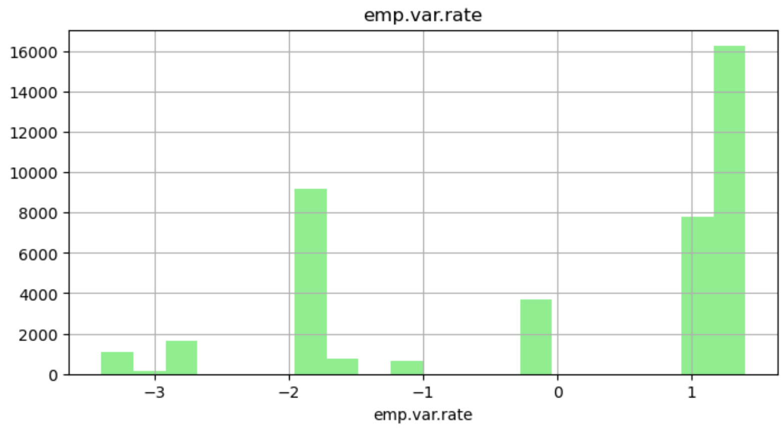


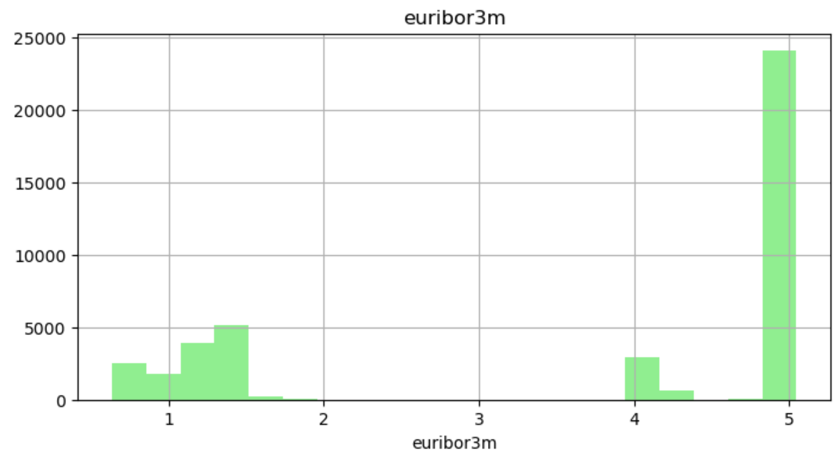
poutcome	y_mean
failure	0.142286
nonexistent	0.088322
success	0.651129

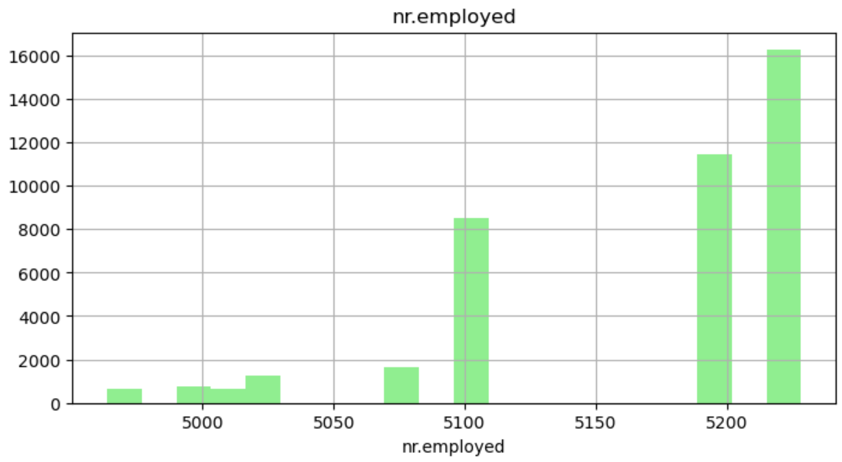
Social and Economic Context Attributes












```
     emp.var.rate
y
no       0.248875
yes     -1.233448

############################################
     cons.price.idx
y
no          93.603757
yes         93.354386

############################################
     cons.conf.idx
y
no       -40.593097
yes      -39.789784

############################################
     euribor3m
y
no    3.811491
yes   2.123135

############################################
     nr.employed
y
no   5176.166600
yes  5095.115991
```

Data Preparation

- Values labeled as "**unknown**" were not deleted. Sensible imputations were carried out, associating **education**, **housing**, and **loan** variables with the job. This approach aligns with the real-world scenario where job is correlated with education, housing, and loan status.

```python
df.loc[(df['age']>65) & (df['job']=='unknown'), 'job'] = 'retired'
```

```python
for job_category in df['job'].unique():
    mode_value = df[df['job'] == job_category]['education'].mode().iloc[0]
    df.loc[(df['job'] == job_category) & (df['education'] == 'unknown'), 'education'] = mode_value


cross_table_after_fill = pd.crosstab(df['job'], df['education'], margins=False)

print(cross_table_after_fill)
```

```python
# Using the precalculated cross-table to fill "unknown" values
def fill_unknown_loan(row):
    if row['loan'] == 'unknown':
        job = row['job']
        total_known_loan = cross_table3.loc[job, ['no', 'yes']].sum()

        # If the total known data count is 0, fill as "no"
        if total_known_loan == 0:
            return 'no'

        # If the total known data count is not zero, fill using the ratio
        probability_no = cross_table3.loc[job, 'no'] / total_known_loan
        probability_yes = 1 - probability_no  # Probability of being "yes"
        return np.random.choice(['no', 'yes'], p=[probability_no, probability_yes])
    else:
        return row['loan']


df['loan'] = df.apply(fill_unknown_loan, axis=1)

print(df['loan'].value_counts())
```

```python
# Using the precalculated cross-table to fill "unknown" values
def fill_unknown_housing(row):
    if row['housing'] == 'unknown':
        job = row['job']
        total_known_housing = cross_table2.loc[job, ['no', 'yes']].sum()

        # If the total known data count is 0, fill as "no"
        if total_known_housing == 0:
            return 'no'

        # If the total known data count is not zero, fill using the ratio
        probability_no = cross_table2.loc[job, 'no'] / total_known_housing
        probability_yes = 1 - probability_no  # Probability of being "yes"
        return np.random.choice(['no', 'yes'], p=[probability_no, probability_yes])
    else:
        return row['housing']


df['housing'] = df.apply(fill_unknown_housing, axis=1)

print(df['housing'].value_counts())
```

```python
for education_category in df['education'].unique():
    if education_category != 'unknown':
        mode_value = df[df['education'] == education_category]['job'].mode().iloc[0]
        df.loc[(df['education'] == education_category) & (df['job'] == 'unknown'), 'job'] = mode_value


cross_table_after_fill2 = pd.crosstab(df['job'], df['education'], margins=False)


print(cross_table_after_fill2)
```

Data Preparation

- **pdays:** number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means the client was not previously contacted)

The variable 'pdays' has been rearranged as follows:

```
# Defining the categories with custom labels
bins = [-1, 0, 10, float('inf')]
labels = ['not.contacted', 'recent.contacted', 'moderate.contacted']

# Creating a new categorical column with direct replacement
df['pdays_category'] = pd.cut(np.where(df['pdays'] == 999, -1, df['pdays']), bins=bins, labels=labels, right=False)

print(df['pdays_category'].value_counts())
```

```
pdays_category
not.contacted         39673
recent.contacted       1259
moderate.contacted      256
Name: count, dtype: int64
```

Modelling

- Due to the imbalance in the dataset, sampling techniques were employed. (Note: Initially, machine learning calculations were conducted without utilizing sampling methods, yielding significantly poor results.)
- Four different machine learning models have been selected: **Logistic Regression, Random Forest, Adaboost** and **XGBoost**.
- The calculated ROC AUC and recall scores after running the models are sorted as follows:
  - **ROC-AUC for Logistic Regression**: 0.7299 **Recall for Logistic Regression**: 0.61
  - **ROC-AUC for Random Forest**: 0.6599 **Recall for Random Forest**: 0.38
  - **ROC-AUC for AdaBoost**: 0.7319 **Recall for AdaBoost**: 0.61
  - **ROC-AUC for XGBoost**: 0.6457 **Recall for XGBoost**: 0.36

- The **Adaboost** model seems to be the most suitable, based on the evaluation of ROC-AUC and Recall metrics.

Note: As mentioned earlier, the results were obtained without applying outlier suppression and excluding unknown values from the dataset to better align with real-world scenarios. While it's possible to improve results by dealing with outliers and removing unknown values, I opted not to follow this approach.

Thank You