eda

December 3, 2024

1 Exploration data analysis

This final project for CSCA5622 Introduction to machine learning: Supervised Learning. We are using data from UCI Machine Learning Repository (UCI Repository). In this opportunity, we are using the dataset called 'Bank Marketing', the data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (variable y).

2 Dataset information

The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

2.1 Variables tables

Variable Name	Role	Type	Demographic	Description	Units	Missing Values
age	Feature	Integer	Age	Age of the client		no
job	Feature	Categorical	Occupation	Type of job		no
marital	Feature	Categorical	Marital Status	Marital status		no
education	Feature	Categorical	Education Level	Education level		no
default	Feature	Binary		Whether the client has credit in default		no
balance	Feature	Integer		Average yearly balance in euros		no
housing	Feature	Binary		Whether the client has a housing loan		no
loan	Feature	Binary		Whether the client has a personal loan		no
contact	Feature	Categorical		Communication type used to contact the client		yes
day_of_week	Feature	Date		Last contact day of the week		no
month	Feature	Date		Last contact month of the year		no
duration	Feature	Integer		Duration of the last contact in seconds (numeric).		no
campaign	Feature	Integer		Number of contacts performed during this campaign for this client		no

Variable Name	Role	Type	Demographic	Description	Units	Missing Values
pdays	Feature	Integer		Number of days since the client was last contacted from a previous campaign		yes
previous	Feature	Integer		Number of contacts performed before this campaign and for this client		no
poutcome	Feature	Categorical		Outcome of the previous marketing campaign (‘failure’, ‘nonexistent’, ‘success’)		yes
y	Target	Binary		Whether the client subscribed to a term deposit		no

2.2 Additional information

2.2.1 Input variables:

Bank client data: - **age** (numeric) - **job** : type of job (categorical: “admin.”,“unknown”,“unemployed”,“management”,“housemaid”,“entrepreneur”,“student”, “blue-collar”,“self-employed”,“retired”,“technician”,“services”) - **marital** : marital status (categorical: “married”,“divorced”,“single”; note: “divorced” means divorced or widowed) - **education** (categorical: “unknown”,“secondary”,“primary”,“tertiary”) - **default**: has credit in default? (binary: “yes”,“no”) - **balance**: average yearly balance, in euros (numeric) - **housing**: has housing loan? (binary: “yes”,“no”) - **loan**: has personal loan? (binary: “yes”,“no”) #### Related with the last contact of the current campaign: - **contact**: contact communication type (categorical: “unknown”,“telephone”,“cellular”) - **day**: last contact day of the month (numeric) - **month**: last contact month of year (categorical: “jan”, “feb”, “mar”, …, “nov”, “dec”) - **duration**: last contact duration, in seconds (numeric) #### Other attributes: - **campaign**: number of contacts performed during this campaign and for this client (numeric, includes last contact) - **pdays**: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted) - **previous**: number of contacts performed before this campaign and for this client (numeric) - **poutcome**: outcome of the previous marketing campaign (categorical: “unknown”,“other”,“failure”,“success”)

2.2.2 Output variable (desired target):

- **y** - has the client subscribed a term deposit? (binary: “yes”,“no”)

3 Project Summary

3.1 Objective

The goal of this project is to build a classification model that predicts whether a customer will subscribe to a term deposit based on various features from the bank’s marketing campaign data. The project will involve exploring and preprocessing the data, selecting relevant features, applying machine learning algorithms, and evaluating model performance using appropriate metrics. The final aim is to develop a robust model that helps the bank optimize its marketing efforts by targeting customers who are most likely to subscribe to a term deposit.

3.2 Import libraries

[24]:
```
from ucimlrepo import fetch_ucirepo
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from utils import *

warnings.filterwarnings('ignore')
pd.set_option('display.max_columns',None)
pd.set_option('display.max_rows',None)


# fetch dataset
bank_marketing = fetch_ucirepo(id=222)

# data (as pandas dataframes)
X = bank_marketing.data.features
y = bank_marketing.data.targets

# metadata
print(bank_marketing.metadata)

# variable information
print(bank_marketing.variables)
```


```
{'uci_id': 222, 'name': 'Bank Marketing', 'repository_url':
'https://archive.ics.uci.edu/dataset/222/bank+marketing', 'data_url':
'https://archive.ics.uci.edu/static/public/222/data.csv', 'abstract': 'The data
is related with direct marketing campaigns (phone calls) of a Portuguese banking
institution. The classification goal is to predict if the client will subscribe
a term deposit (variable y).', 'area': 'Business', 'tasks': ['Classification'],
'characteristics': ['Multivariate'], 'num_instances': 45211, 'num_features': 16,
'feature_types': ['Categorical', 'Integer'], 'demographics': ['Age',
'Occupation', 'Marital Status', 'Education Level'], 'target_col': ['y'],
'index_col': None, 'has_missing_values': 'yes', 'missing_values_symbol': 'NaN',
'year_of_dataset_creation': 2014, 'last_updated': 'Fri Aug 18 2023',
'dataset_doi': '10.24432/C5K306', 'creators': ['S. Moro', 'P. Rita', 'P.
Cortez'], 'intro_paper': {'ID': 277, 'type': 'NATIVE', 'title': 'A data-driven
approach to predict the success of bank telemarketing', 'authors': 'Sérgio Moro,
P. Cortez, P. Rita', 'venue': 'Decision Support Systems', 'year': 2014,
'journal': None, 'DOI': '10.1016/j.dss.2014.03.001', 'URL': 'https://www.semanti
cscholar.org/paper/cab86052882d126d43f72108c6cb41b295cc8a9e', 'sha': None,
'corpus': None, 'arxiv': None, 'mag': None, 'acl': None, 'pmid': None, 'pmcid':
None}, 'additional_info': {'summary': "The data is related with direct marketing
campaigns of a Portuguese banking institution. The marketing campaigns were
```

based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed. \n\nThere are four datasets: \n1) bank-additional-full.csv with all examples (41188) and 20 inputs, ordered by date (from May 2008 to November 2010), very close to the data analyzed in [Moro et al., 2014]\n2) bank-additional.csv with 10% of the examples (4119), randomly selected from 1), and 20 inputs.\n3) bank-full.csv with all examples and 17 inputs, ordered by date (older version of this dataset with less inputs). \n4) bank.csv with 10% of the examples and 17 inputs, randomly selected from 3 (older version of this dataset with less inputs). \nThe smallest datasets are provided to test more computationally demanding machine learning algorithms (e.g., SVM). \n\nThe classification goal is to predict if the client will subscribe (yes/no) a term deposit (variable y).", 'purpose': None, 'funded_by': None, 'instances_represent': None, 'recommended_data_splits': None, 'sensitive_data': None, 'preprocessing_description': None, 'variable_info': 'Input variables:\n # bank client data:\n   1 - age (numeric)\n   2 - job : type of job (categorical: "admin.","unknown","unemployed","management","housemaid","entrepreneur","student",\n   "blue-collar","self-employed","retired","technician","services") \n   3 - marital : marital status (categorical: "married","divorced","single"; note: "divorced" means divorced or widowed)\n   4 - education (categorical: "unknown","secondary","primary","tertiary")\n   5 - default: has credit in default? (binary: "yes","no")\n   6 - balance: average yearly balance, in euros (numeric) \n   7 - housing: has housing loan? (binary: "yes","no")\n   8 - loan: has personal loan? (binary: "yes","no")\n   # related with the last contact of the current campaign:\n   9 - contact: contact communication type (categorical: "unknown","telephone","cellular") \n 10 - day: last contact day of the month (numeric)\n 11 - month: last contact month of year (categorical: "jan", "feb", "mar", …, "nov", "dec")\n 12 - duration: last contact duration, in seconds (numeric)\n   # other attributes:\n 13 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)\n 14 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)\n 15 - previous: number of contacts performed before this campaign and for this client (numeric)\n 16 - poutcome: outcome of the previous marketing campaign (categorical: "unknown","other","failure","success")\n\n Output variable (desired target):\n 17 - y - has the client subscribed a term deposit? (binary: "yes","no")\n', 'citation': None}}

```
          name     role         type      demographic  \
0          age  Feature      Integer              Age
1          job  Feature  Categorical       Occupation
2      marital  Feature  Categorical   Marital Status
3    education  Feature  Categorical  Education Level
4      default  Feature       Binary             None
5      balance  Feature      Integer             None
6      housing  Feature       Binary             None
7         loan  Feature       Binary             None
8      contact  Feature  Categorical             None
```

```
9   day_of_week  Feature         Date            None
10        month  Feature         Date            None
11     duration  Feature      Integer            None
12     campaign  Feature      Integer            None
13        pdays  Feature      Integer            None
14     previous  Feature      Integer            None
15     poutcome  Feature  Categorical            None
16            y   Target       Binary            None

                                          description  units missing_values
0                                                None   None             no
1   type of job (categorical: 'admin.','blue-colla…   None             no
2   marital status (categorical: 'divorced','marri…   None             no
3   (categorical: 'basic.4y','basic.6y','basic.9y'…   None             no
4                           has credit in default?   None             no
5                           average yearly balance  euros             no
6                                has housing loan?   None             no
7                               has personal loan?   None             no
8   contact communication type (categorical: 'cell…   None            yes
9                       last contact day of the week   None             no
10  last contact month of year (categorical: 'jan'…   None             no
11   last contact duration, in seconds (numeric). …   None             no
12  number of contacts performed during this campa…   None             no
13  number of days that passed by after the client…   None            yes
14  number of contacts performed before this campa…   None             no
15  outcome of the previous marketing campaign (ca…   None            yes
16          has the client subscribed a term deposit?   None             no
```

[25]:
```python
# Join the feature dataset and the target dataset to start the exploratory data
↪analysis
df = X.copy()
df['y'] = y
df.sample(3)
```

[25]:
```
       age            job  marital education default  balance housing loan  \
38259   45  self-employed  married  tertiary      no     -497     yes  yes
15526   35   entrepreneur   single  tertiary      no      145     yes   no
42082   61        retired  married   primary      no     8729      no   no

        contact  day_of_week month  duration  campaign  pdays  previous  \
38259  cellular           15   may       217         1    176         1
15526  cellular           18   jul       799         2     -1         0
42082  cellular           30   oct       480         1     -1         0

      poutcome    y
38259  failure   no
15526      NaN  yes
```

```
42082      NaN  yes
```

4 Validation of null values and duplicates rows

[26]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   age          45211 non-null  int64
 1   job          44923 non-null  object
 2   marital      45211 non-null  object
 3   education    43354 non-null  object
 4   default      45211 non-null  object
 5   balance      45211 non-null  int64
 6   housing      45211 non-null  object
 7   loan         45211 non-null  object
 8   contact      32191 non-null  object
 9   day_of_week  45211 non-null  int64
 10  month        45211 non-null  object
 11  duration     45211 non-null  int64
 12  campaign     45211 non-null  int64
 13  pdays        45211 non-null  int64
 14  previous     45211 non-null  int64
 15  poutcome     8252 non-null   object
 16  y            45211 non-null  object
dtypes: int64(7), object(10)
memory usage: 5.9+ MB
```

[27]:
```python
#Percentage of null values per column

print(df.isna().sum()/df.shape[0]*100)
```

```
age             0.000000
job             0.637013
marital         0.000000
education       4.107407
default         0.000000
balance         0.000000
housing         0.000000
loan            0.000000
contact        28.798301
day_of_week     0.000000
month           0.000000
duration        0.000000
```

```
campaign         0.000000
pdays            0.000000
previous         0.000000
poutcome        81.747805
y                0.000000
dtype: float64
```

We observe that in the dataset, the variables with the highest number of null values are the following:

- poutcome : 81.74%
- contact : 28.79
- education : 4.10%
- job : 0.63%

Therefore, we must define the strategy according to the context of each variable. For the categorical variables, poutcome, contact, education we will fill the empty fields with the category "unknown", since the variable definition allows us to use that category when the data is unknown. For the job variable, we can have two strategies, either we delete the rows with empty fields, or we fill the fields with the fashion of the category, since the percentage of null values is very small, we will perform the second strategy.

[28]:
```
df['poutcome'] = df['poutcome'].fillna('unknown')
df['contact'] = df['contact'].fillna('unknown')
df['education'] = df['education'].fillna('unknown')
df['job'] = df['job'].fillna(df['job'].mode().values[0])
```

[29]:
```
print(df.isna().sum()/df.shape[0]*100)
```

```
age            0.0
job            0.0
marital        0.0
education      0.0
default        0.0
balance        0.0
housing        0.0
loan           0.0
contact        0.0
day_of_week    0.0
month          0.0
duration       0.0
campaign       0.0
pdays          0.0
previous       0.0
poutcome       0.0
y              0.0
dtype: float64
```

[30]:
```
# Selecting duplicate rows except first
# occurrence based on all columns
```

```
duplicate = df[df.duplicated()]

print("Duplicate Rows :")

# Print the resultant Dataframe
duplicate
```

```
Duplicate Rows :
```

[30]:
```
Empty DataFrame
Columns: [age, job, marital, education, default, balance, housing, loan,
contact, day_of_week, month, duration, campaign, pdays, previous, poutcome, y]
Index: []
```

We don't have duplicate values in the dataset. So we don't need to drop rows in this case.

5 Statistical Analysis

[31]:
```
df.describe().T
```

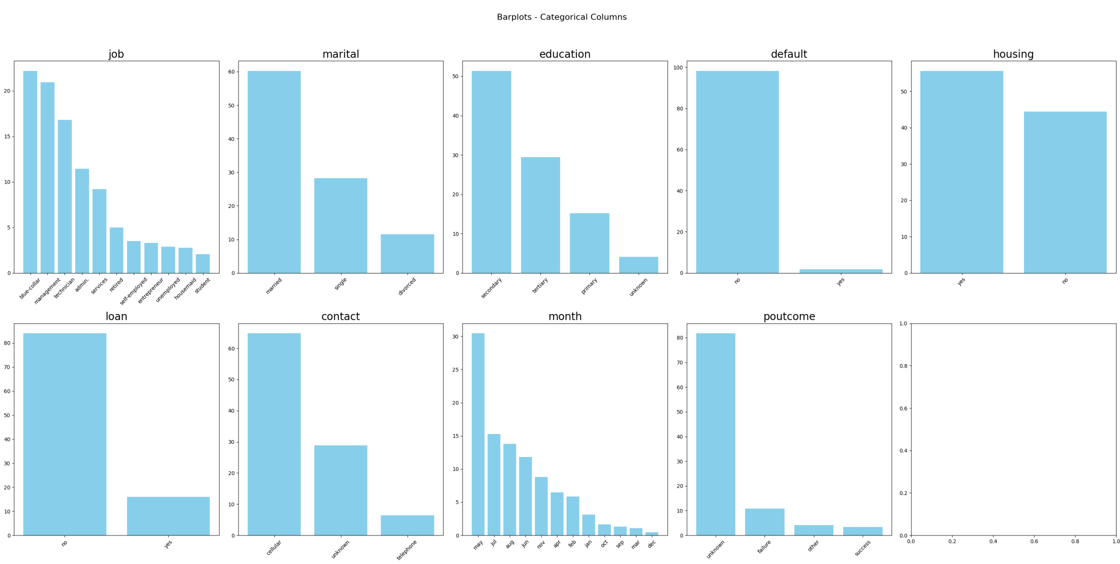
[31]:
```
               count         mean          std      min    25%    50%     75%  \
age          45211.0    40.936210    10.618762     18.0   33.0   39.0    48.0
balance      45211.0  1362.272058  3044.765829  -8019.0   72.0  448.0  1428.0
day_of_week  45211.0    15.806419     8.322476      1.0    8.0   16.0    21.0
duration     45211.0   258.163080   257.527812      0.0  103.0  180.0   319.0
campaign     45211.0     2.763841     3.098021      1.0    1.0    2.0     3.0
pdays        45211.0    40.197828   100.128746     -1.0   -1.0   -1.0    -1.0
previous     45211.0     0.580323     2.303441      0.0    0.0    0.0     0.0

                  max
age              95.0
balance      102127.0
day_of_week      31.0
duration       4918.0
campaign         63.0
pdays           871.0
previous        275.0
```

6 Univariable analysis

6.1 Categorical variables

[32]:
```
# Categorical variables
categorical_columns =␣
↪['job','marital','education','default','housing','loan','contact','month','poutcome']

# Create subplots for each variable
```

```python
fig, axes = plt.subplots(nrows=2, ncols=5, figsize=(30, 15), sharey=False)

# Flatten the axes array for easier indexing
axes = axes.flatten()

# Plot each categorical column
for i, column in enumerate(categorical_columns):
    percentage = df[column].value_counts(normalize=True) * 100
    percentage_df = percentage.reset_index()
    percentage_df.columns = [column, 'Percentage']

    axes[i].bar(percentage_df[column], percentage_df['Percentage'], color='skyblue')
    axes[i].set_title(column, fontsize=20)
    axes[i].tick_params(axis='x', rotation=45)  # Rotate x-axis labels for better visibility

# Add a main title
fig.suptitle("Barplots - Categorical Columns", fontsize=16)

# Adjust layout to prevent overlap
plt.tight_layout(rect=[0, 0, 1, 0.95])

# Show the plot
plt.show()
```



[33]:
```python
df['y'].value_counts(normalize=True)*100
```

[33]:
```
y
no     88.30152
yes    11.69848
Name: proportion, dtype: float64
```

[34]:
```
df['default'].value_counts(normalize=True)*100
```

[34]:
```
default
no     98.197341
yes     1.802659
Name: proportion, dtype: float64
```

On this occasion, we observed that the "default" variable has very little variability (98% no, 1.8% yes). Therefore, we see that the information provided by the categorical variable is not very significant, and we could even consider deleting the variable when training the model. Therefore, we are going to delete this variable from the final dataset.

6.2 Numerical variables

6.3 Outliers

[35]:
```
numerical_columns = df.select_dtypes([np.number]).columns.tolist()
print(numerical_columns)
numeric_statistics(data=df,numeric_columns=numerical_columns)
```

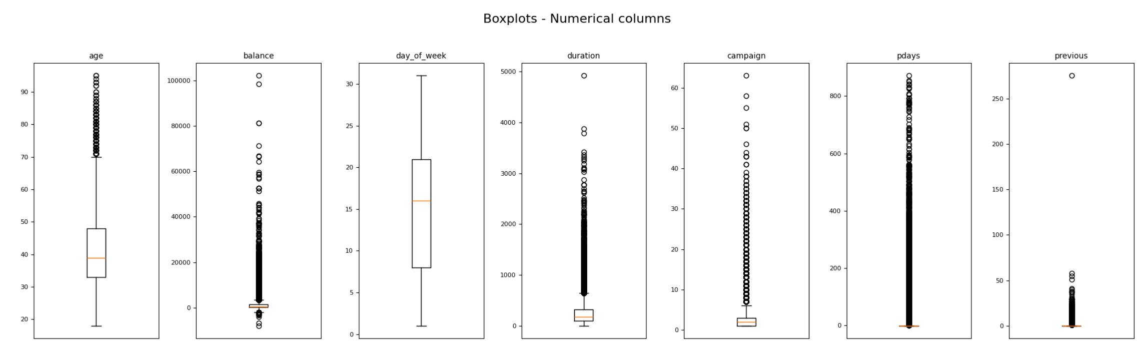
```
['age', 'balance', 'day_of_week', 'duration', 'campaign', 'pdays', 'previous']
```

[35]:
```
      Variable  Median         Mean  Std. Deviation  Min. Value  Percentile 5  \
0          age    39.0    40.936210       10.618762          18          27.0
1      balance   448.0  1362.272058     3044.765829       -8019        -172.0
2  day_of_week    16.0    15.806419        8.322476           1           3.0
3     duration   180.0   258.163080      257.527812           0          35.0
4     campaign     2.0     2.763841        3.098021           1           1.0
5        pdays    -1.0    40.197828      100.128746          -1          -1.0
6     previous     0.0     0.580323        2.303441           0           0.0

   Percentile 10  Percentile 15  Percentile 20  Percentile 25  Percentile 50  \
0           29.0           30.0           32.0           33.0           39.0
1            0.0            0.0           22.0           72.0          448.0
2            5.0            6.0            7.0            8.0           16.0
3           58.0           75.0           89.0          103.0          180.0
4            1.0            1.0            1.0            1.0            2.0
5           -1.0           -1.0           -1.0           -1.0           -1.0
6            0.0            0.0            0.0            0.0            0.0

   Percentile 75  Percentile 80  Percentile 85  Percentile 90  Percentile 95  \
0           48.0           51.0           53.0           56.0           59.0
1         1428.0         1859.0         2539.0         3574.0         5768.0
2           21.0           24.0           27.0           28.0           29.0
```

```
3          319.0          368.0          437.0          548.0          751.0
4            3.0            4.0            4.0            5.0            8.0
5           -1.0           -1.0          102.0          185.0          317.0
6            0.0            0.0            1.0            2.0            3.0

   Percentile 99  Max. value
0           71.0          95
1        13164.9      102127
2           31.0          31
3         1269.0        4918
4           16.0          63
5          370.0         871
6            8.9         275
```

[36]:

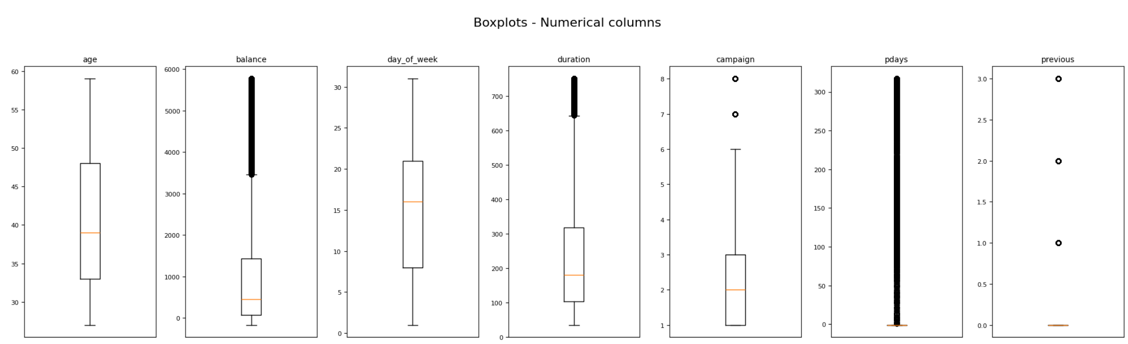
```python
# Create subplots for each variable on its own scale
fig, axes = plt.subplots(nrows=1, ncols=7, figsize=(20, 6), sharey=False)

# Plot each column separately
for i, column in enumerate(numerical_columns):
    axes[i].boxplot(df[column], vert=True)
    axes[i].set_title(column, fontsize=10)
    axes[i].tick_params(axis='x', which='both', bottom=False, top=False,␣
 ↪labelbottom=False)
    axes[i].tick_params(axis='y', labelsize=8)

# Add a main title
fig.suptitle("Boxplots - Numerical columns", fontsize=16)
plt.tight_layout(rect=[0, 0, 1, 0.95])

# Show the plot
plt.show()
```



We can observe that in the variables age, balance, duration, campaign, pdays, previous we have outliers to identify. If we do not refine these values, we will not be able to have an optimal model that generalizes the results when tested with other populations or samples. On this occasion, we are going to use the winzor approach.

[37]:
```
from scipy.stats.mstats import winsorize

outlier_columns = ['age','balance','duration','campaign','pdays','previous']
for col in outlier_columns:
    data = df[col].values
    winzorized_data = winsorize(data, limits=[0.05, 0.05])
    df[col] = winzorized_data
```

[38]:
```
# Create subplots for each variable on its own scale
fig, axes = plt.subplots(nrows=1, ncols=7, figsize=(20, 6), sharey=False)

# Plot each column separately
for i, column in enumerate(numerical_columns):
    axes[i].boxplot(df[column], vert=True)
    axes[i].set_title(column, fontsize=10)
    axes[i].tick_params(axis='x', which='both', bottom=False, top=False,␣
 ↪labelbottom=False)
    axes[i].tick_params(axis='y', labelsize=8)

# Add a main title
fig.suptitle("Boxplots - Numerical columns", fontsize=16)
plt.tight_layout(rect=[0, 0, 1, 0.95])

# Show the plot
plt.show()
```

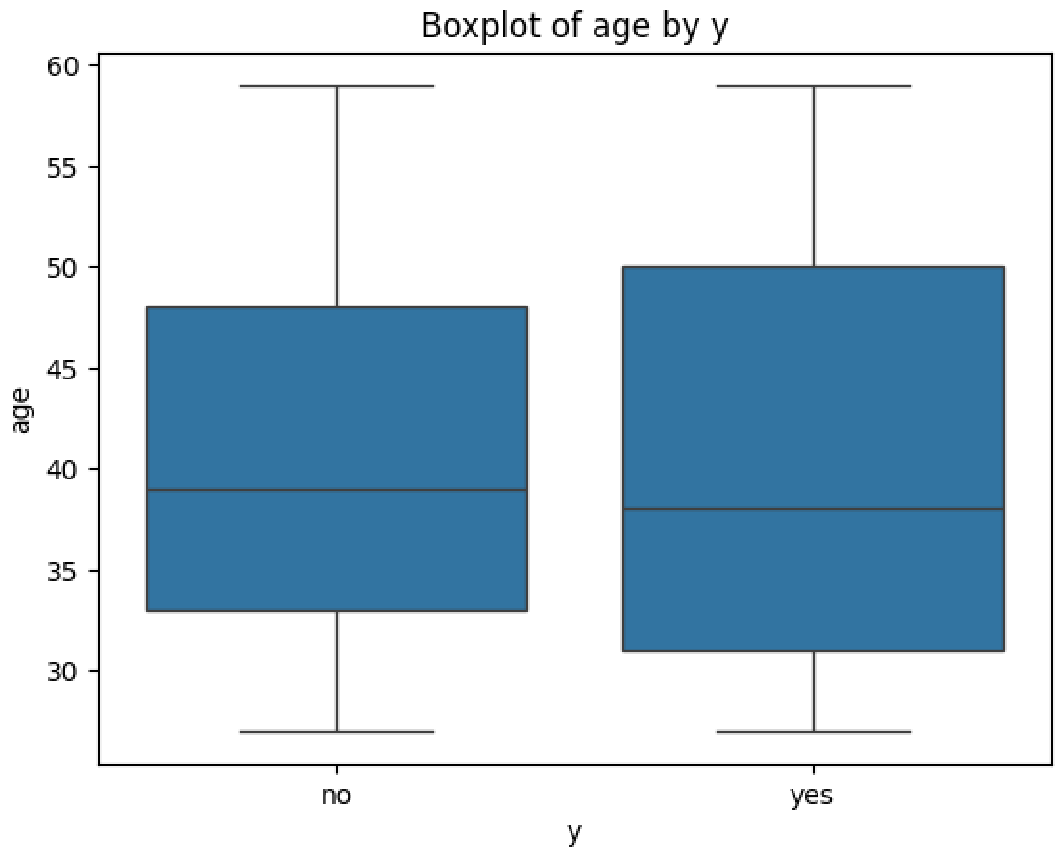


7 Correlation Analysis

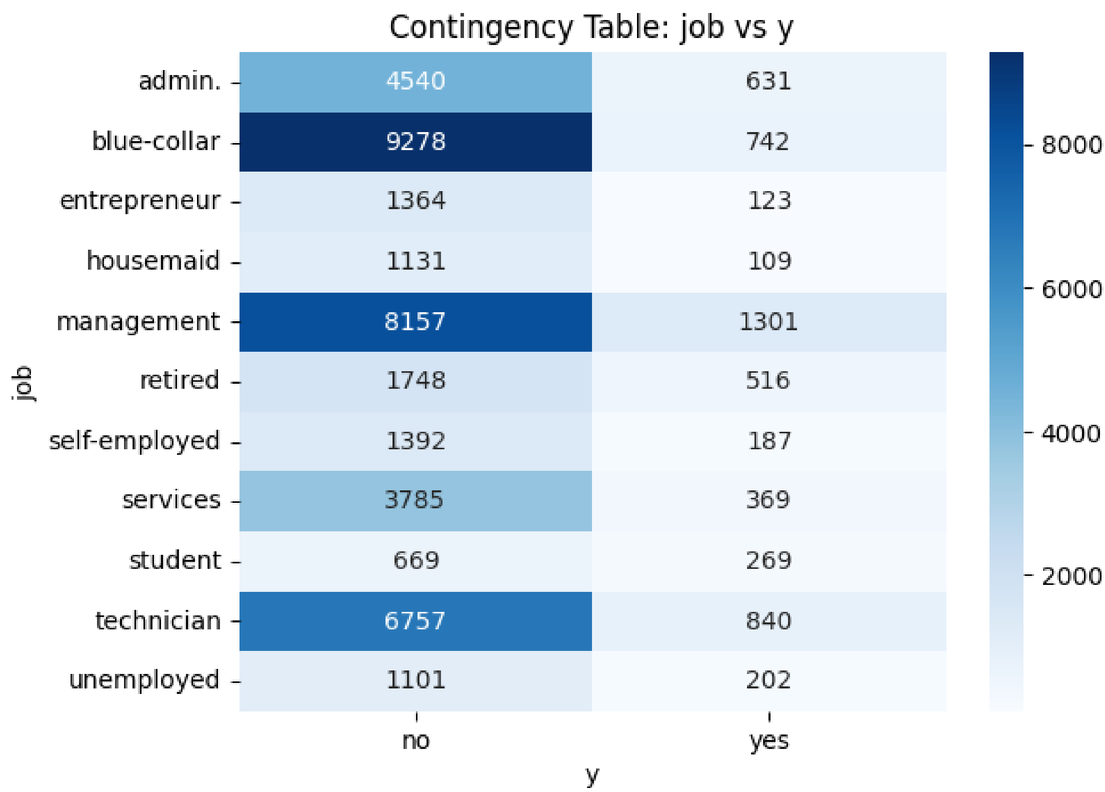
In this part, we start with the correlation analysis between the categorical and numerical variables with the target variable (**'y'**). To determine the correlation and interaction between the variables of the model and the target variable, with this we will be able to define which variables we will finally take in the model, since we will need to consider the variables that have more information and interaction with the target variable. Subsequently, we will review the interaction between numerical and categorical variables, to validate that there are no variables that give us a high interaction and may cause the model to take redundant variables.

[39]:
```
relationship_with_target(df=df)
```

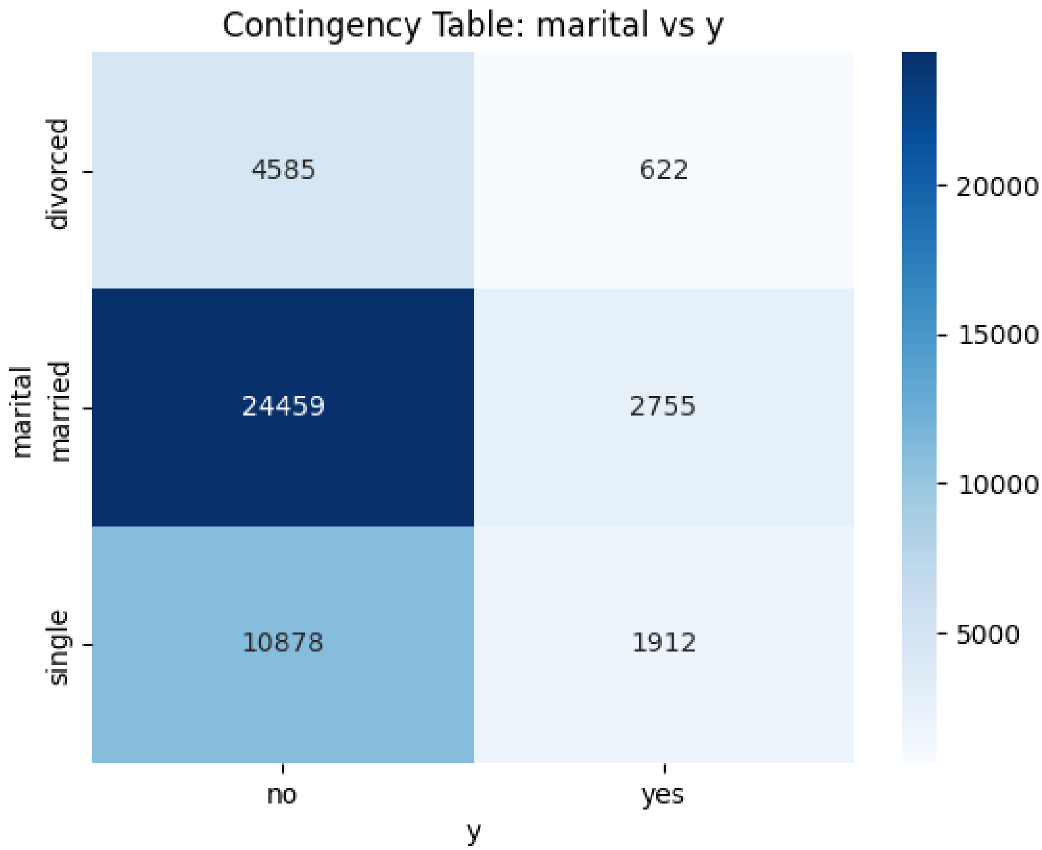
```
Analyzing age vs Target (y):
```



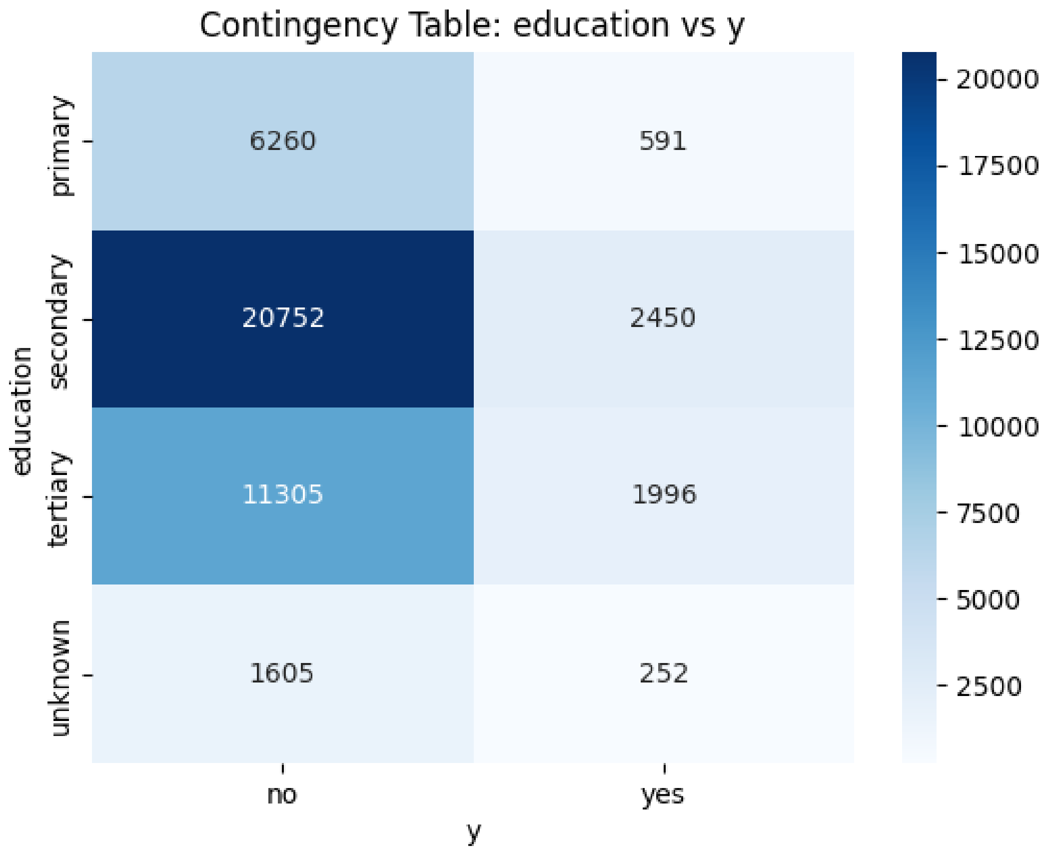

```
y=no: Mean=40.75, Std=9.51
y=yes: Mean=40.88, Std=10.99


Analyzing job vs Target (y):
Chi-square Test between job and y: p-value = 5.575427995540736e-172
```



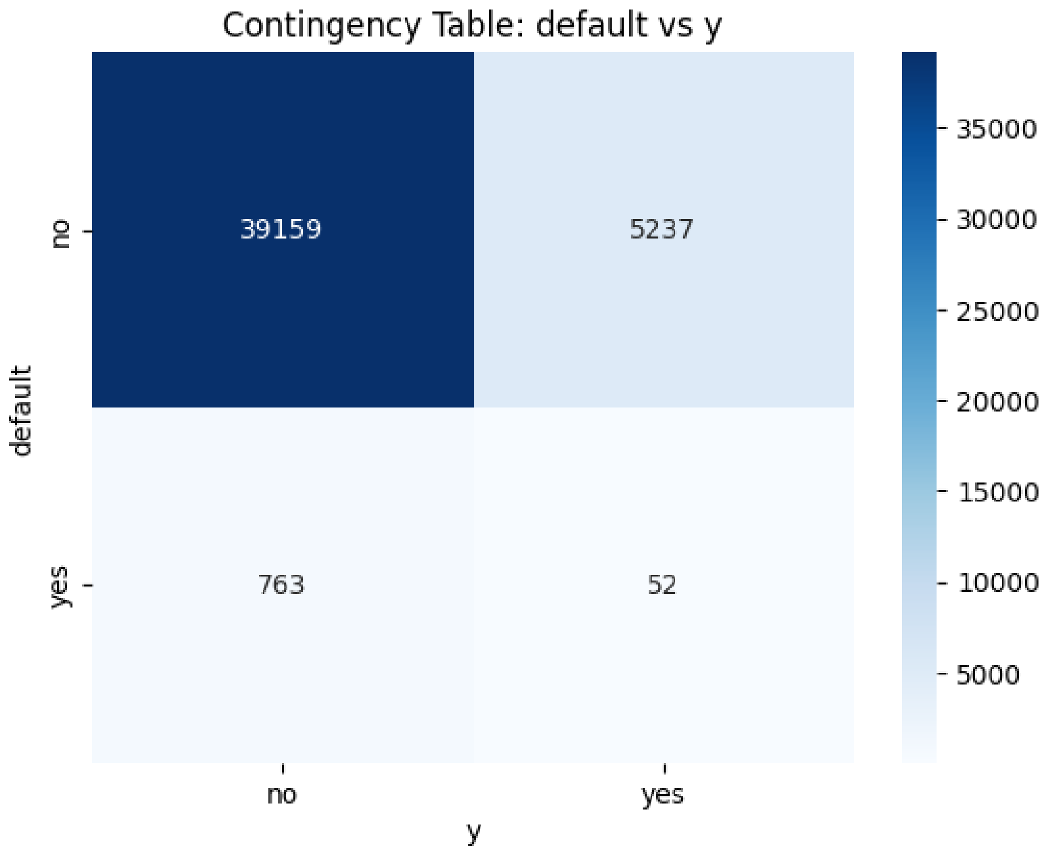

```
Analyzing marital vs Target (y):
Chi-square Test between marital and y: p-value = 2.1450999986791792e-43
```



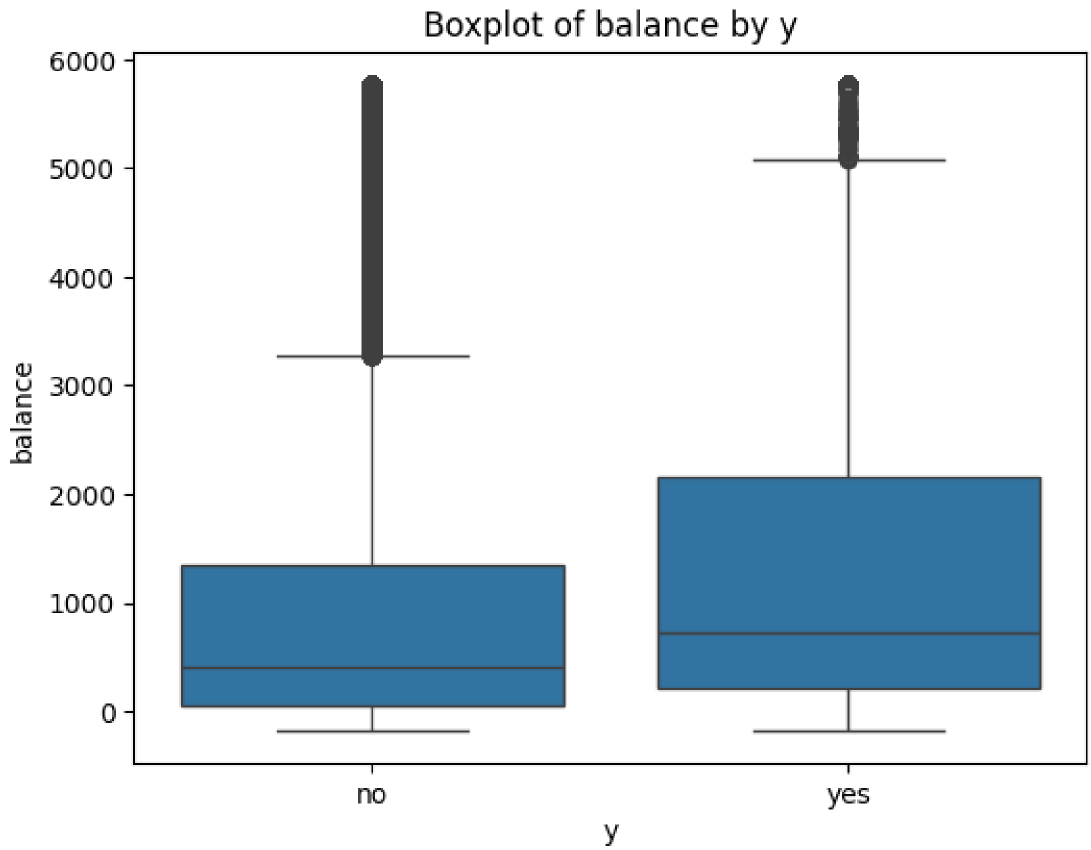

```
Analyzing education vs Target (y):
Chi-square Test between education and y: p-value = 1.6266562124072994e-51
```



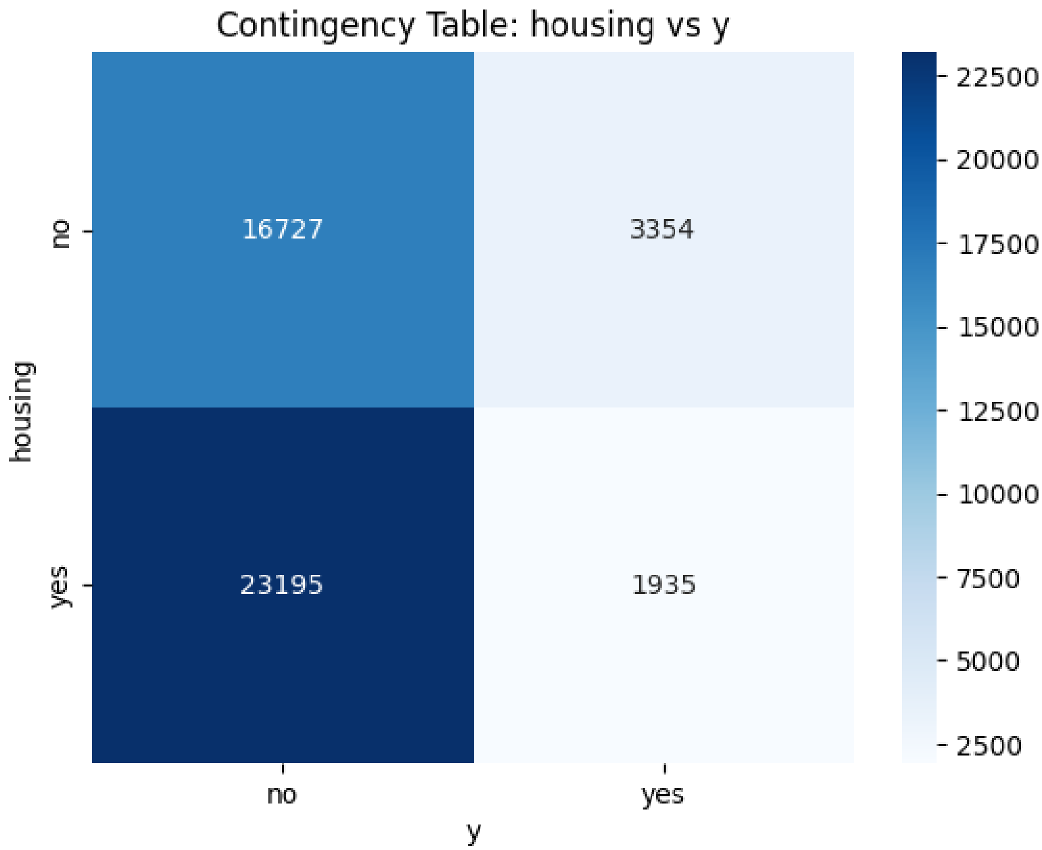

```
Analyzing default vs Target (y):
Chi-square Test between default and y: p-value = 2.4538606753508344e-06
```



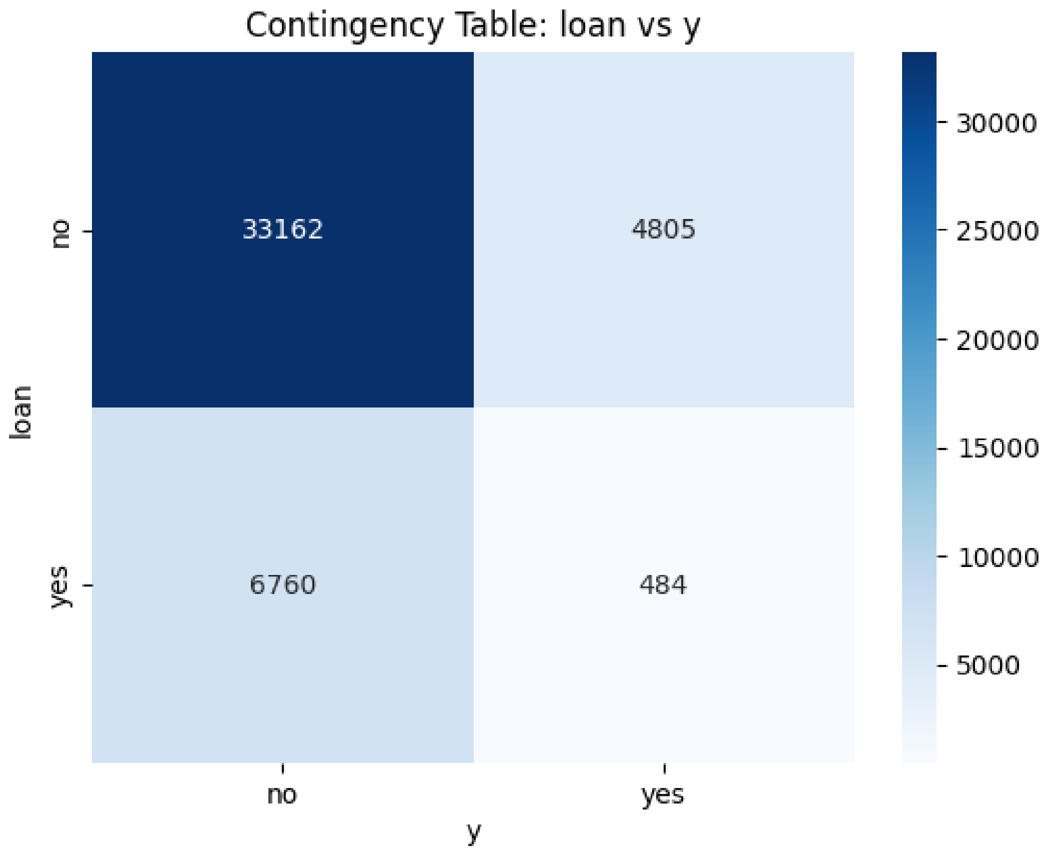
Analyzing balance vs Target (y):



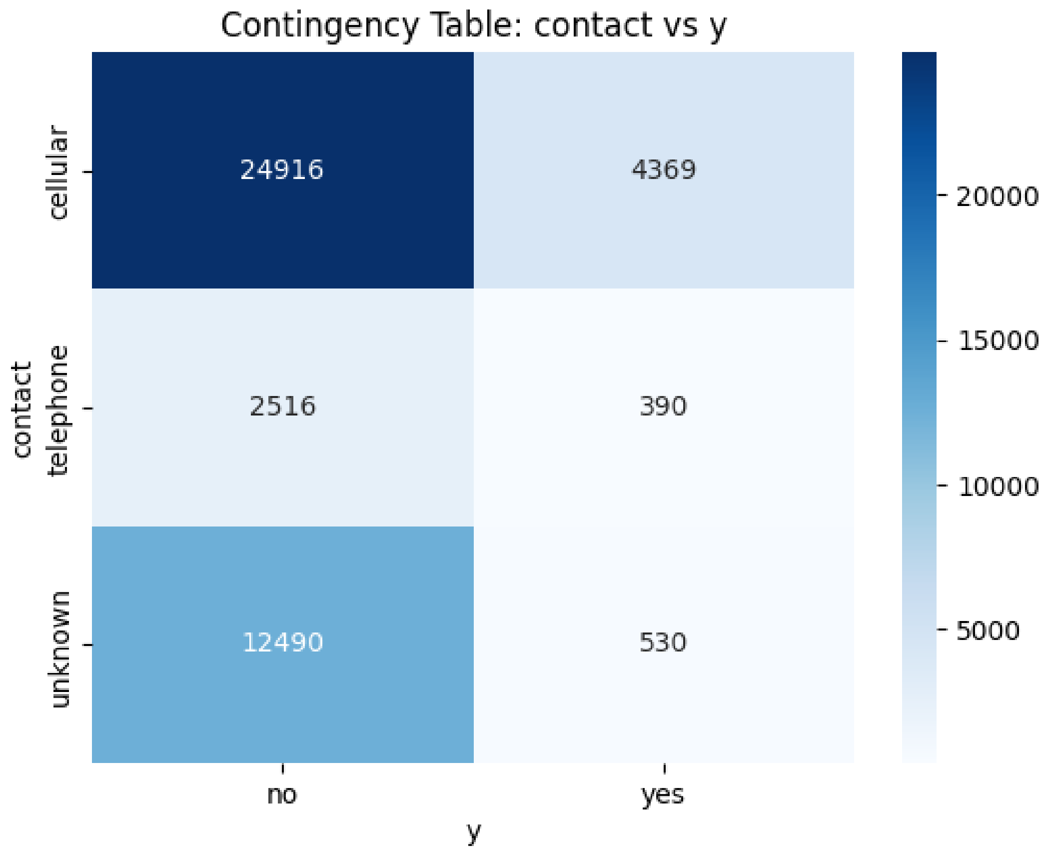

```
y=no: Mean=1070.83, Std=1556.99
y=yes: Mean=1465.41, Std=1714.48


Analyzing housing vs Target (y):
Chi-square Test between housing and y: p-value = 2.918797605076633e-192
```



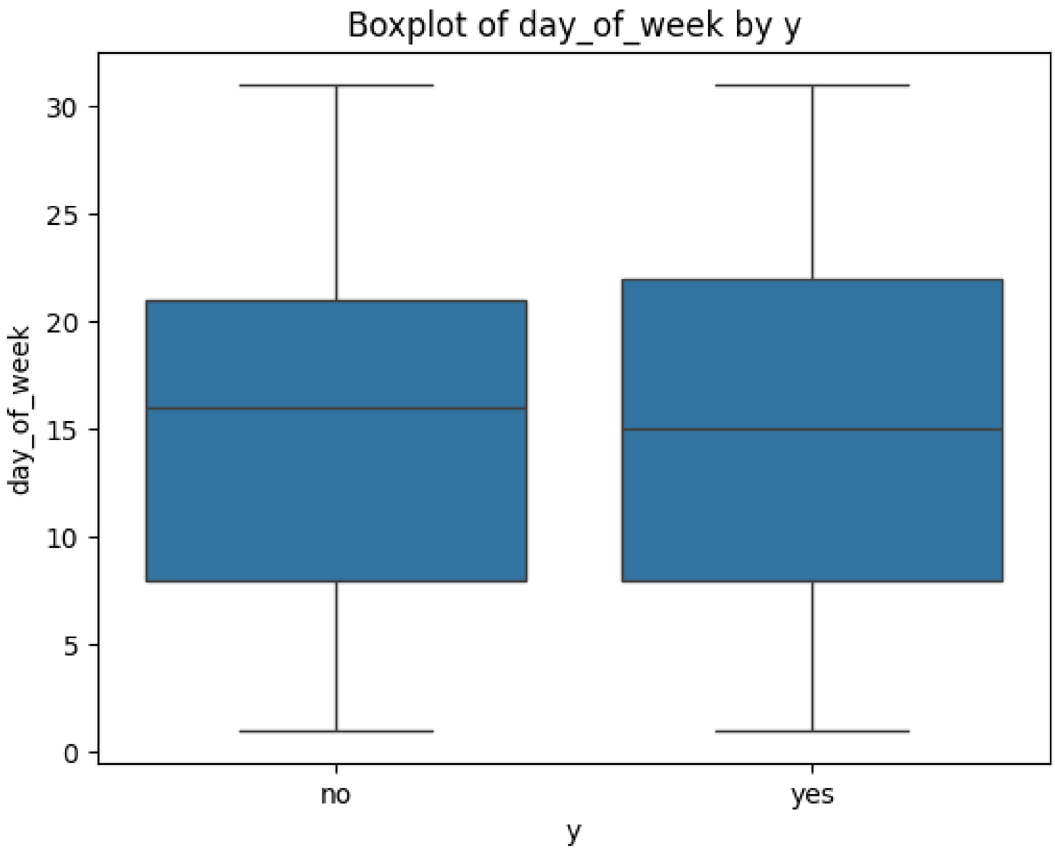

```
Analyzing loan vs Target (y):
Chi-square Test between loan and y: p-value = 1.665061163492756e-47
```




```
Analyzing contact vs Target (y):
Chi-square Test between contact and y: p-value = 1.251738325340638e-225
```

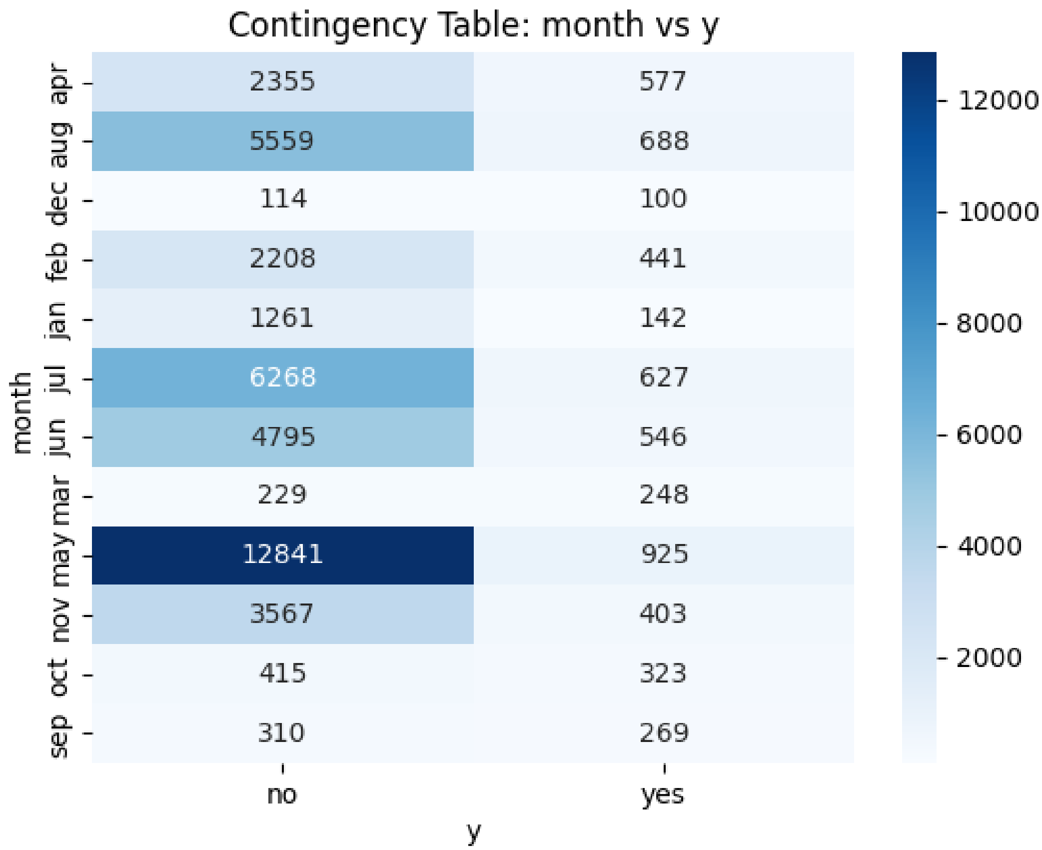


Analyzing day_of_week vs Target (y):

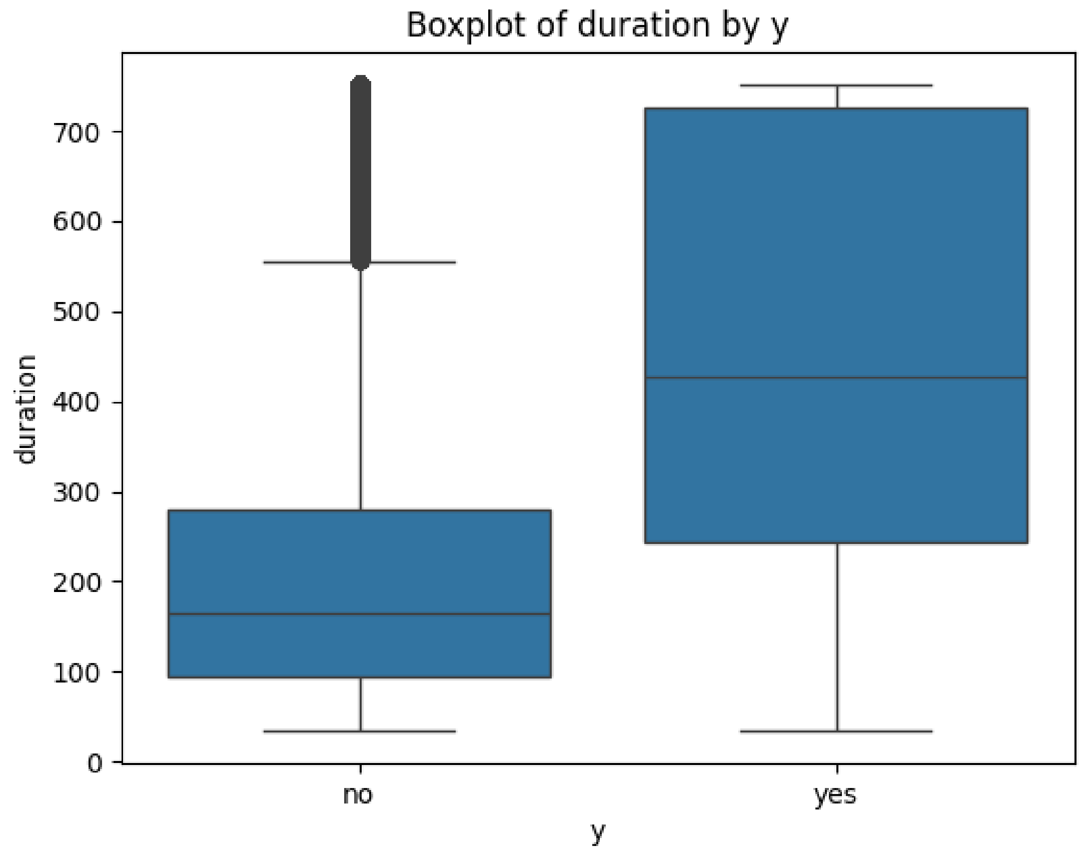



```
y=no: Mean=15.89, Std=8.29
y=yes: Mean=15.16, Std=8.50
```

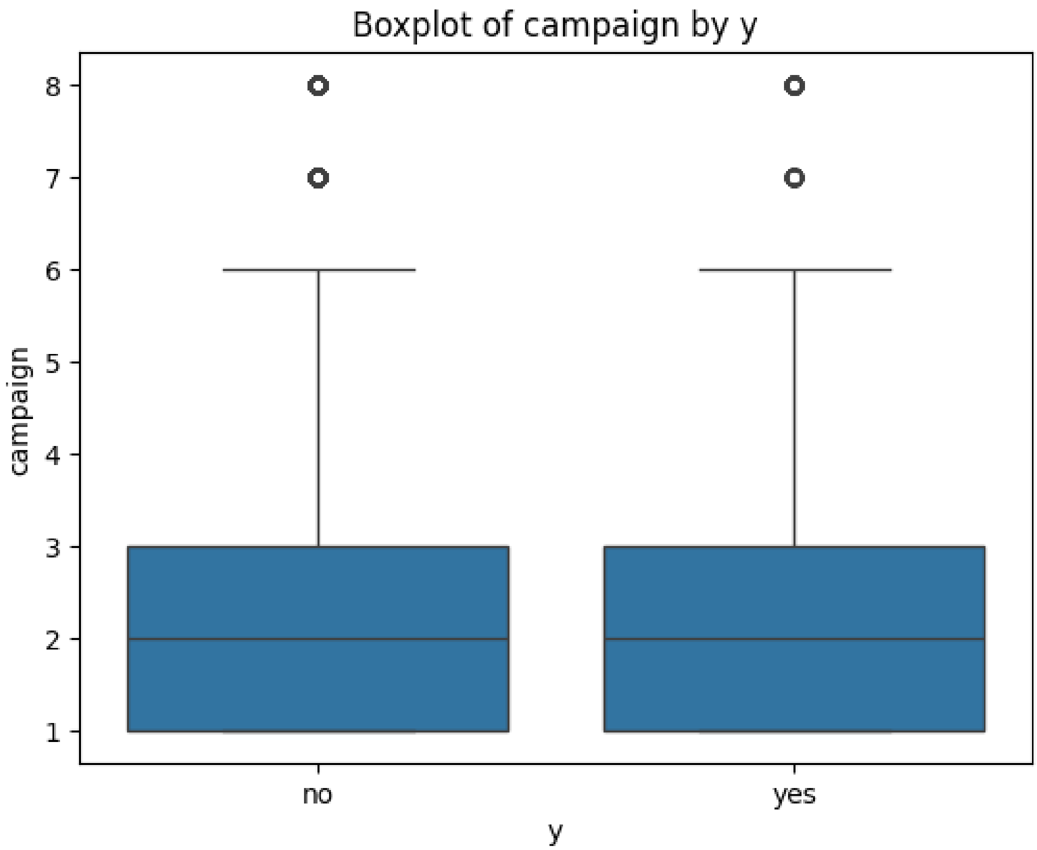
```
Analyzing month vs Target (y):
Chi-square Test between month and y: p-value = 0.0
```



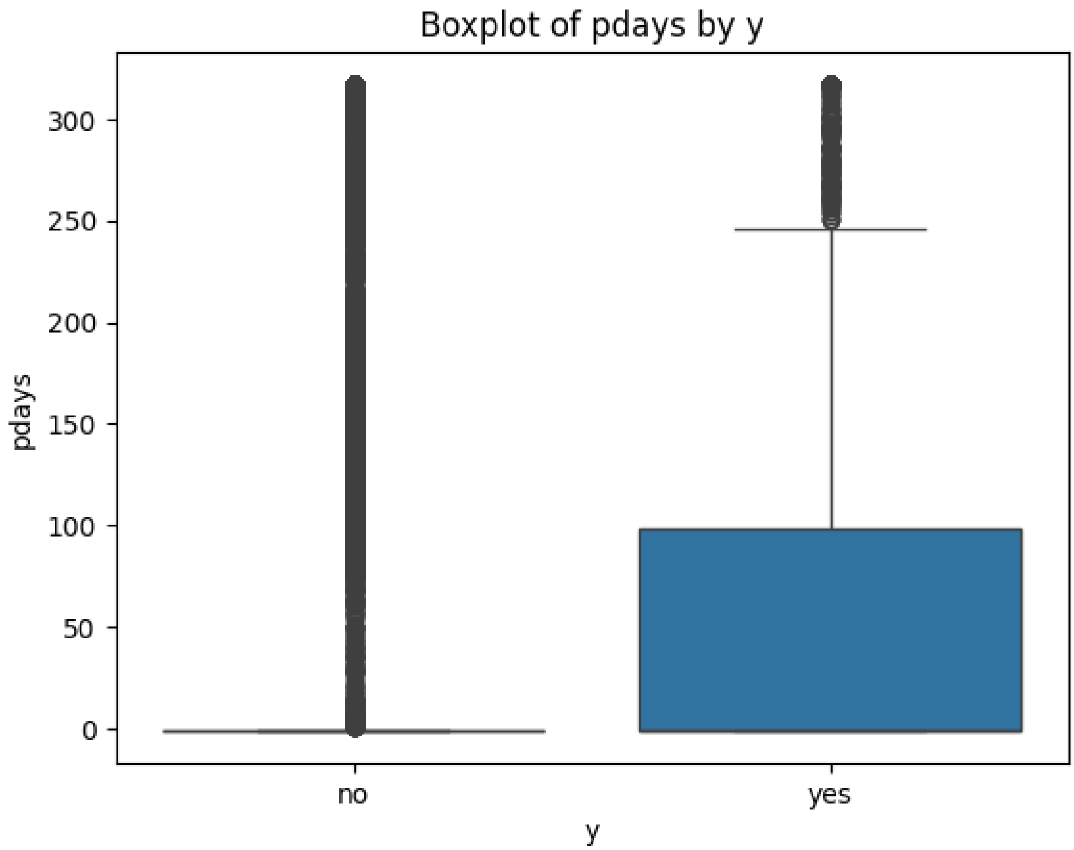

```
Analyzing duration vs Target (y):
```



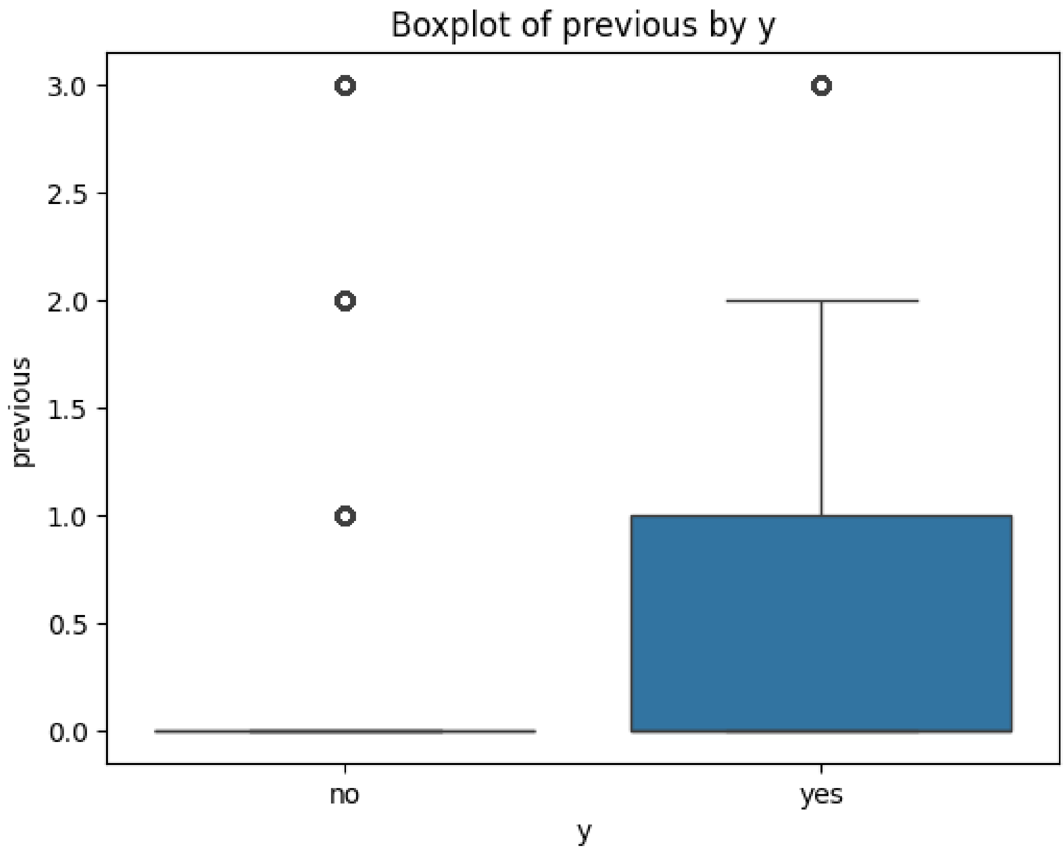

```
y=no: Mean=214.18, Std=167.40
y=yes: Mean=453.89, Std=228.83


Analyzing campaign vs Target (y):
```



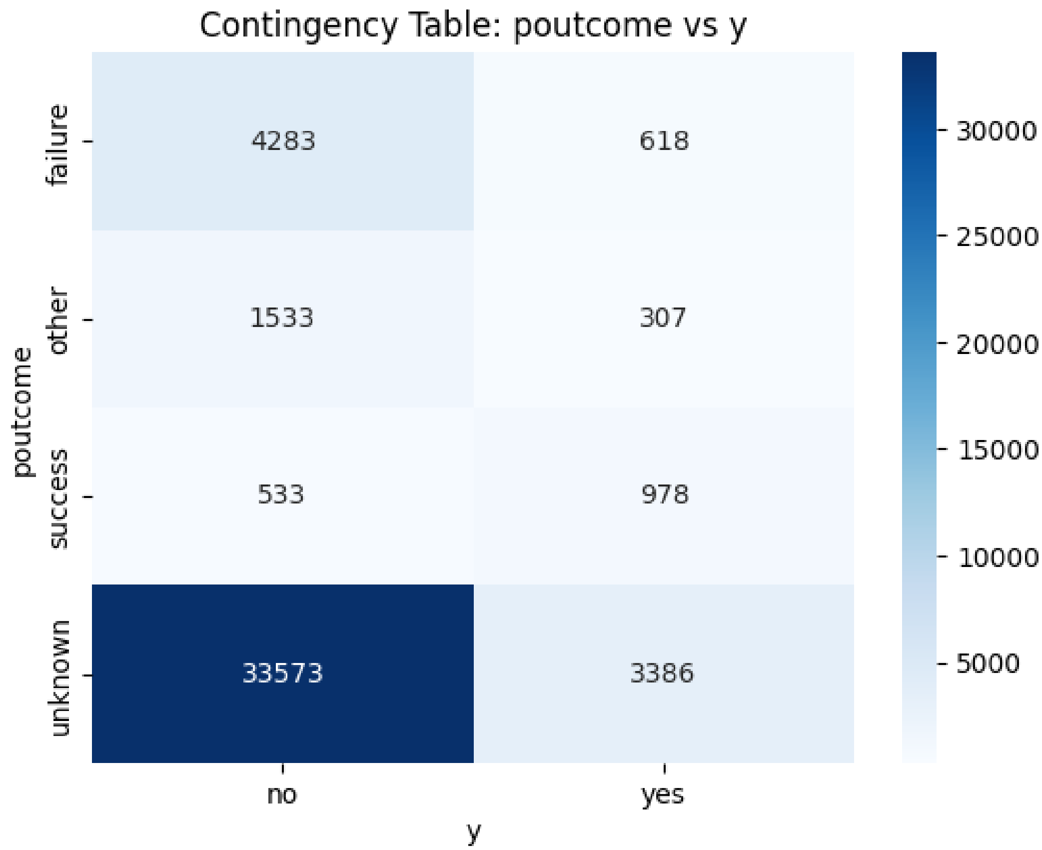

```
y=no: Mean=2.57, Std=1.94
y=yes: Mean=2.07, Std=1.53


Analyzing pdays vs Target (y):
```




```
y=no: Mean=34.20, Std=88.50
y=yes: Mean=62.89, Std=99.87


Analyzing previous vs Target (y):
```

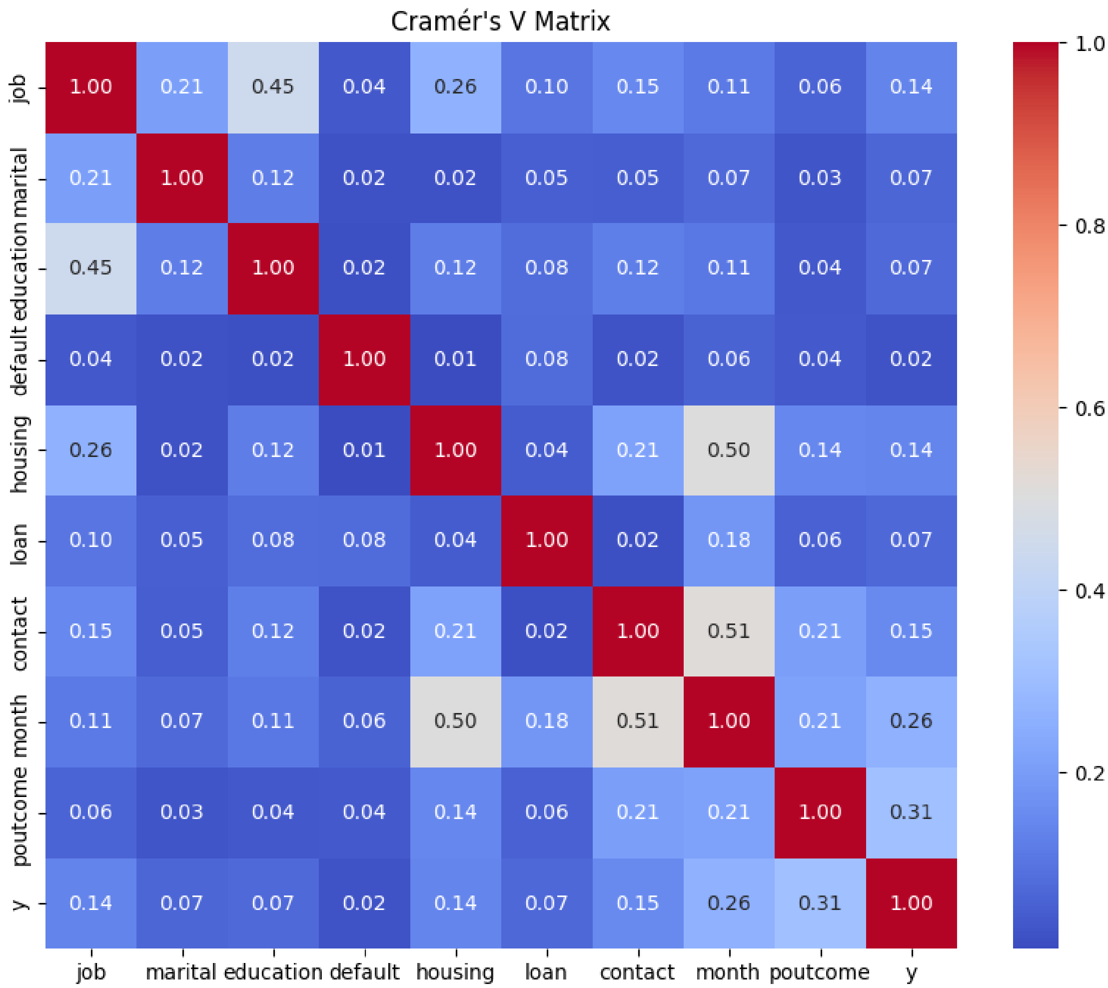



```
y=no: Mean=0.33, Std=0.82
y=yes: Mean=0.77, Std=1.15


Analyzing poutcome vs Target (y):
Chi-square Test between poutcome and y: p-value = 0.0
```



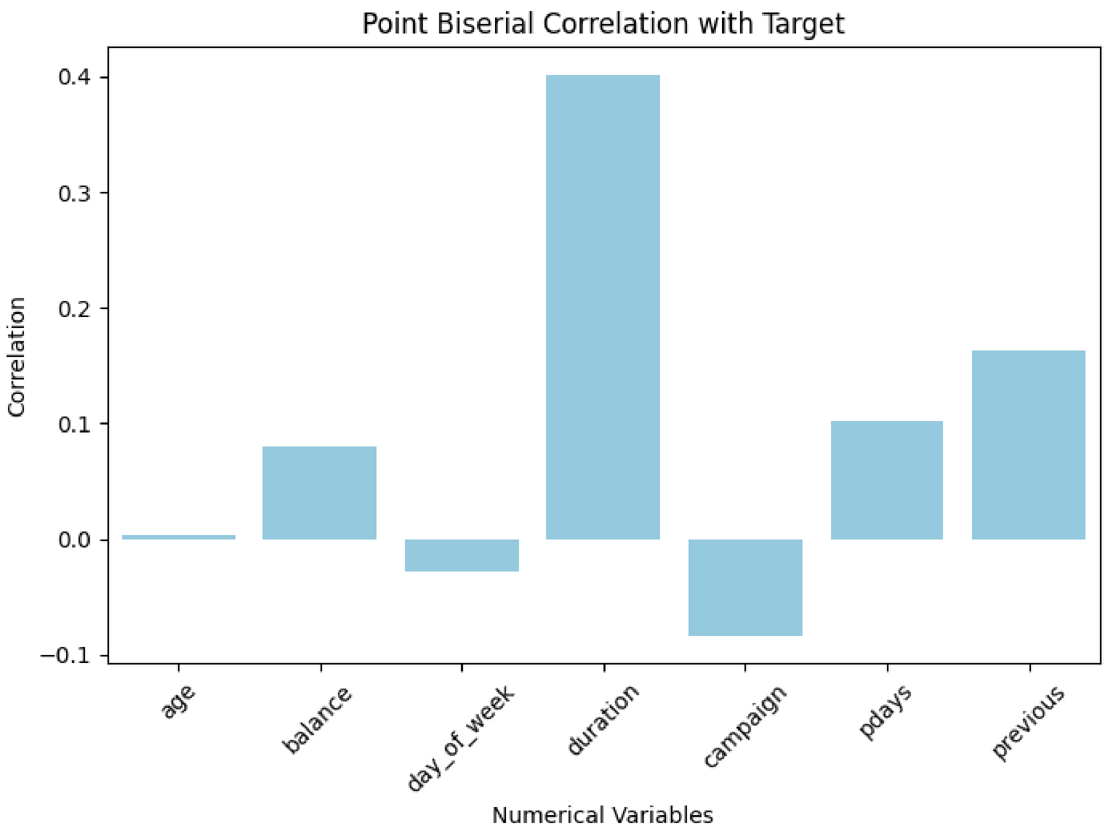
With the observed results, we conclude that the "default" variable does not influence the "y" objective, so it may be detrimental to add a variable that does not contribute relevant information to the model. For the numerical variables we observe that in all variables there is a difference in the distribution between the categories of the target variable "y". Therefore, we will keep all the numerical variables for the final dataset of the model.

[40]:
```
# Compute Cramér's V matrix
cramers_matrix = cramers_v_matrix(df, categorical_columns + ['y'])
plt.figure(figsize=(10, 8))
sns.heatmap(cramers_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Cramér's V Matrix")
plt.show()
```

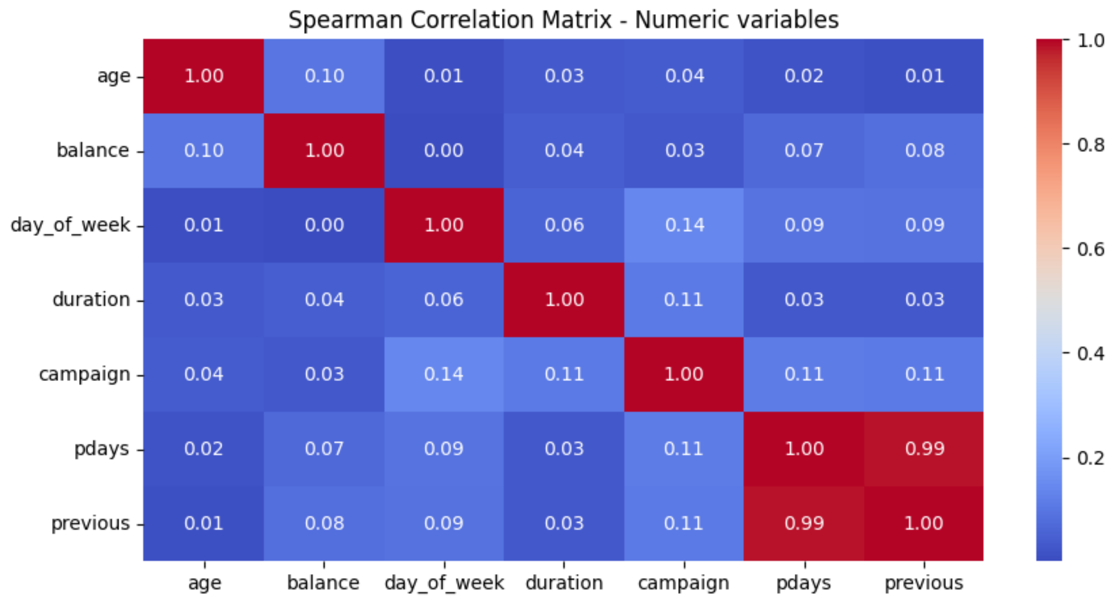


In the cramer's matrix, we note that the categorical variables do not have a strong interaction between them, however we observe that the categorical variable poutcome, month are the variables with the highest correlation with the target 'y'.

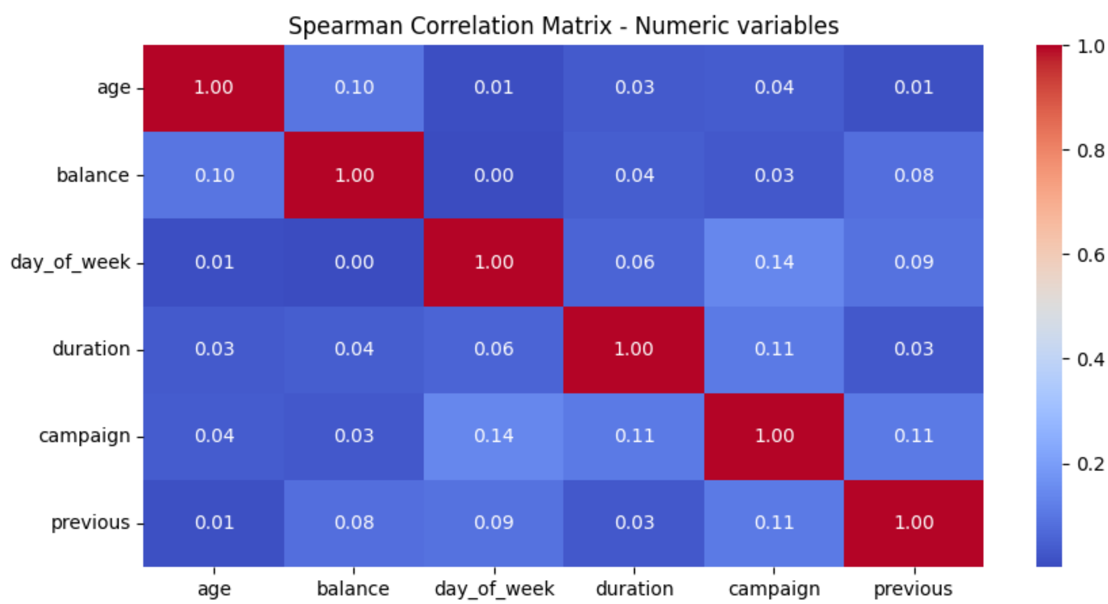
[41]:
```python
# Compute the matrix
biserial_matrix = point_biserial_matrix(df, numerical_columns, 'y')
plt.figure(figsize=(8, 5))
sns.barplot(x=biserial_matrix.index, y="Point Biserial Correlation",␣
 ↪data=biserial_matrix.reset_index(), color='skyblue')
plt.xticks(rotation=45)
plt.title("Point Biserial Correlation with Target")
plt.ylabel("Correlation")
plt.xlabel("Numerical Variables")
plt.show()
```



[42]:
```
spearman_matrix = df[numerical_columns].corr(method='spearman').abs()
fig = plt.figure(figsize=(10,5))
sns.heatmap(spearman_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title('Spearman Correlation Matrix - Numeric variables')
plt.show()
```



[43]:
```
numerical_columns.remove('pdays')
spearman_matrix = df[numerical_columns].corr(method='spearman').abs()
fig = plt.figure(figsize=(10,5))
sns.heatmap(spearman_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title('Spearman Correlation Matrix - Numeric variables')
plt.show()
```



In the spearman correlation matrix between numerical variables, we notice that there is a very strong correlation between the variables ‘pdays’ and ‘previous’, which indicates that these variables together in a model can be counterproductive in the training of a model. Therefore, we can consider deleting one of the variables.

Due to the results of the interaction analysis of each of the available variables with the target, we decided to delete the variable pdays because it maintains a very strong interaction with the target (check point biserial correlation with target).

[44]:
```
# Final dataset - Next step train model
final_columns = list(set(categorical_columns)) + list(set(numerical_columns)) +␣
↪['y']
print(final_columns)
final_dataset = df[final_columns].copy()
final_dataset.to_csv('./data/final_data.csv',sep='|',index=None)
```

```
['contact', 'housing', 'poutcome', 'job', 'loan', 'month', 'marital', 'default',
'education', 'age', 'previous', 'balance', 'day_of_week', 'campaign',
'duration', 'y']
```