notebook

June 12, 2025

1 1. Exploratory Data Analysis (EDA)

1.1 1.1. Setup

First, let's import the necessary libraries for data manipulation and visualization and load the dataset.

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

1.2 1.2. Initial Data Inspection

Let's get a first look at the data's structure, content, and statistical summary. For that define the paths:

[2]:
```python
BANK_PATH = "data/bank/bank.csv"
BANK_FULL_PATH = "data/bank/bank-full.csv"
BANK_ADDITIONAL_PATH = "data/bank-additional/bank-additional.csv"
BANK_ADDITIONAL_FULL_PATH = "data/bank-additional/bank-additional-full.csv"
```

[3]:
```python
bank_df = pd.read_csv(BANK_PATH, sep=";")
bank_full_df = pd.read_csv(BANK_FULL_PATH,sep=";")
bank_additional_df = pd.read_csv(BANK_ADDITIONAL_PATH,sep=";")
bank_additional_full_df = pd.read_csv(BANK_ADDITIONAL_FULL_PATH,sep=";")
```

[4]:
```python
bank_df
```

[4]:
```
      age            job  marital  education default  balance housing loan  \
0      30     unemployed  married    primary      no     1787      no   no
1      33       services  married  secondary      no     4789     yes  yes
2      35     management   single   tertiary      no     1350     yes   no
3      30     management  married   tertiary      no     1476     yes  yes
4      59    blue-collar  married  secondary      no        0     yes   no
…     …              …        …          …       …        …       …    …
4516   33       services  married  secondary      no     -333     yes   no
4517   57  self-employed  married   tertiary     yes    -3313     yes  yes
4518   57     technician  married  secondary      no      295      no   no
```

```
4519   28     blue-collar  married  secondary      no     1137      no   no
4520   44    entrepreneur   single   tertiary      no     1136     yes  yes

       contact  day month  duration  campaign  pdays  previous poutcome   y
0     cellular   19   oct        79         1     -1         0  unknown  no
1     cellular   11   may       220         1    339         4  failure  no
2     cellular   16   apr       185         1    330         1  failure  no
3      unknown    3   jun       199         4     -1         0  unknown  no
4      unknown    5   may       226         1     -1         0  unknown  no
…          …  …    …      …         …   …       …        …  ..
4516  cellular   30   jul       329         5     -1         0  unknown  no
4517   unknown    9   may       153         1     -1         0  unknown  no
4518  cellular   19   aug       151        11     -1         0  unknown  no
4519  cellular    6   feb       129         4    211         3    other  no
4520  cellular    3   apr       345         2    249         7    other  no

[4521 rows x 17 columns]
```

[5]:
```
bank_full_df
```

[5]:
```
       age           job   marital  education default  balance housing loan  \
0       58    management   married   tertiary      no     2143     yes   no
1       44    technician    single  secondary      no       29     yes   no
2       33  entrepreneur   married  secondary      no        2     yes  yes
3       47   blue-collar   married    unknown      no     1506     yes   no
4       33       unknown    single    unknown      no        1      no   no
…     …           …      …        …      …       …      …   …
45206   51    technician   married   tertiary      no      825      no   no
45207   71       retired  divorced    primary      no     1729      no   no
45208   72       retired   married  secondary      no     5715      no   no
45209   57   blue-collar   married  secondary      no      668      no   no
45210   37  entrepreneur   married  secondary      no     2971      no   no

         contact  day month  duration  campaign  pdays  previous poutcome    y
0        unknown    5   may       261         1     -1         0  unknown   no
1        unknown    5   may       151         1     -1         0  unknown   no
2        unknown    5   may        76         1     -1         0  unknown   no
3        unknown    5   may        92         1     -1         0  unknown   no
4        unknown    5   may       198         1     -1         0  unknown   no
…            …  …    …      …         …   …        …       …   …
45206   cellular   17   nov       977         3     -1         0  unknown  yes
45207   cellular   17   nov       456         2     -1         0  unknown  yes
45208   cellular   17   nov      1127         5    184         3  success  yes
45209  telephone   17   nov       508         4     -1         0  unknown   no
45210   cellular   17   nov       361         2    188        11    other   no

[45211 rows x 17 columns]
```

[6]:
```
bank_additional_df
```

[6]:
```
      age          job  marital          education default  housing     loan  \
0      30  blue-collar  married           basic.9y      no      yes       no
1      39     services   single        high.school      no       no       no
2      25     services  married        high.school      no      yes       no
3      38     services  married           basic.9y      no  unknown  unknown
4      47       admin.  married  university.degree      no      yes       no
…    …           …        …                  …     …       …       …
4114   30       admin.  married           basic.6y      no      yes      yes
4115   39       admin.  married        high.school      no      yes       no
4116   27      student   single        high.school      no       no       no
4117   58       admin.  married        high.school      no       no       no
4118   34   management   single        high.school      no      yes       no

        contact month day_of_week  …  campaign  pdays  previous  \
0      cellular   may         fri  …         2    999         0
1     telephone   may         fri  …         4    999         0
2     telephone   jun         wed  …         1    999         0
3     telephone   jun         fri  …         3    999         0
4      cellular   nov         mon  …         1    999         0
…          …   …          …   …       …    …        …
4114   cellular   jul         thu  …         1    999         0
4115  telephone   jul         fri  …         1    999         0
4116   cellular   may         mon  …         2    999         1
4117   cellular   aug         fri  …         1    999         0
4118   cellular   nov         wed  …         1    999         0

         poutcome emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  \
0     nonexistent         -1.8          92.893          -46.2      1.313
1     nonexistent          1.1          93.994          -36.4      4.855
2     nonexistent          1.4          94.465          -41.8      4.962
3     nonexistent          1.4          94.465          -41.8      4.959
4     nonexistent         -0.1          93.200          -42.0      4.191
…             …          …             …             …        …
4114  nonexistent          1.4          93.918          -42.7      4.958
4115  nonexistent          1.4          93.918          -42.7      4.959
4116      failure         -1.8          92.893          -46.2      1.354
4117  nonexistent          1.4          93.444          -36.1      4.966
4118  nonexistent         -0.1          93.200          -42.0      4.120

      nr.employed   y
0          5099.1  no
1          5191.0  no
2          5228.1  no
3          5228.1  no
4          5195.8  no
```

```
…          …  ..
4114       5228.1  no
4115       5228.1  no
4116       5099.1  no
4117       5228.1  no
4118       5195.8  no

[4119 rows x 21 columns]
```

[7]:
```
bank_additional_full_df
```

[7]:
```
       age          job  marital            education  default housing loan  \
0       56    housemaid  married             basic.4y       no      no   no
1       57     services  married          high.school  unknown      no   no
2       37     services  married          high.school       no     yes   no
3       40       admin.  married             basic.6y       no      no   no
4       56     services  married          high.school       no      no  yes
…     …          …       …                  …      …      …  …
41183   73      retired  married  professional.course       no     yes   no
41184   46  blue-collar  married  professional.course       no      no   no
41185   56      retired  married    university.degree       no     yes   no
41186   44   technician  married  professional.course       no      no   no
41187   74      retired  married  professional.course       no     yes   no

         contact month day_of_week  …  campaign  pdays  previous  \
0      telephone   may         mon  …         1    999         0
1      telephone   may         mon  …         1    999         0
2      telephone   may         mon  …         1    999         0
3      telephone   may         mon  …         1    999         0
4      telephone   may         mon  …         1    999         0
…          …    …           …   …       …     …        …
41183   cellular   nov         fri  …         1    999         0
41184   cellular   nov         fri  …         1    999         0
41185   cellular   nov         fri  …         2    999         0
41186   cellular   nov         fri  …         1    999         0
41187   cellular   nov         fri  …         3    999         1

          poutcome emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  \
0      nonexistent          1.1          93.994          -36.4      4.857
1      nonexistent          1.1          93.994          -36.4      4.857
2      nonexistent          1.1          93.994          -36.4      4.857
3      nonexistent          1.1          93.994          -36.4      4.857
4      nonexistent          1.1          93.994          -36.4      4.857
…              …          …             …              …        …
41183  nonexistent         -1.1          94.767          -50.8      1.028
41184  nonexistent         -1.1          94.767          -50.8      1.028
41185  nonexistent         -1.1          94.767          -50.8      1.028
```

```
41186  nonexistent           -1.1           94.767           -50.8       1.028
41187      failure           -1.1           94.767           -50.8       1.028

       nr.employed    y
0           5191.0   no
1           5191.0   no
2           5191.0   no
3           5191.0   no
4           5191.0   no
…              …   …
41183       4963.6  yes
41184       4963.6   no
41185       4963.6   no
41186       4963.6  yes
41187       4963.6   no

[41188 rows x 21 columns]
```

[8]:

```python
df_list = [bank_df, bank_full_df, bank_additional_df, bank_additional_full_df]
print(f"The bank-full.csv dataset has the shape of: {bank_full_df.shape}")
print(f"It contains the columns: {list(bank_full_df.columns)}\n")
print("-----------------------------------------")
print(f"The bank.csv has the shape of: {bank_df.shape}")
print(f"It contains the columns: {list(bank_df.columns)}\n")
print("-----------------------------------------")
print(f"The bank-additional-full.csv dataset has the shape of:␣
  ↪{bank_additional_full_df.shape}")
print(f"It contains the columns: {list(bank_additional_full_df.columns)}\n")
print("-----------------------------------------")
print(f"The bank-additional.csv dataset has the shape of: {bank_additional_df.
  ↪shape}")
print(f"It contains the columns: {list(bank_additional_df.columns)}\n")
```

```
The bank-full.csv dataset has the shape of: (45211, 17)
It contains the columns: ['age', 'job', 'marital', 'education', 'default',
'balance', 'housing', 'loan', 'contact', 'day', 'month', 'duration', 'campaign',
'pdays', 'previous', 'poutcome', 'y']

-----------------------------------------
The bank.csv has the shape of: (4521, 17)
It contains the columns: ['age', 'job', 'marital', 'education', 'default',
'balance', 'housing', 'loan', 'contact', 'day', 'month', 'duration', 'campaign',
'pdays', 'previous', 'poutcome', 'y']

-----------------------------------------
The bank-additional-full.csv dataset has the shape of: (41188, 21)
It contains the columns: ['age', 'job', 'marital', 'education', 'default',
'housing', 'loan', 'contact', 'month', 'day_of_week', 'duration', 'campaign',
```

```
'pdays', 'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx',
'cons.conf.idx', 'euribor3m', 'nr.employed', 'y']

----------------------------------------
The bank-additional.csv dataset has the shape of: (4119, 21)
It contains the columns: ['age', 'job', 'marital', 'education', 'default',
'housing', 'loan', 'contact', 'month', 'day_of_week', 'duration', 'campaign',
'pdays', 'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx',
'cons.conf.idx', 'euribor3m', 'nr.employed', 'y']
```

Verification Conclusion: The shapes match the documentation, and the smaller files are confirmed to be true subsets of the larger files. We can now confidently proceed with `bank-additional-full.csv` for our analysis.

[18]:
```python
# get full statistical metrics on numerical columns
bank_additional_full_df.describe()
```

[18]:

	age	duration	campaign	pdays	previous
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000
mean	40.02406	258.285010	2.567593	962.475454	0.172963
std	10.42125	259.279249	2.770014	186.910907	0.494901
min	17.00000	0.000000	1.000000	0.000000	0.000000
25%	32.00000	102.000000	1.000000	999.000000	0.000000
50%	38.00000	180.000000	2.000000	999.000000	0.000000
75%	47.00000	319.000000	3.000000	999.000000	0.000000
max	98.00000	4918.000000	56.000000	999.000000	7.000000

	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000
mean	0.081886	93.575664	-40.502600	3.621291	5167.035911
std	1.570960	0.578840	4.628198	1.734447	72.251528
min	-3.400000	92.201000	-50.800000	0.634000	4963.600000
25%	-1.800000	93.075000	-42.700000	1.344000	5099.100000
50%	1.100000	93.749000	-41.800000	4.857000	5191.000000
75%	1.400000	93.994000	-36.400000	4.961000	5228.100000
max	1.400000	94.767000	-26.900000	5.045000	5228.100000

1.3 1.3. Data Cleaning

We'll check for any missing values and duplicates. For that we will need to see what column entries exists for each column in the first place.

[9]:
```python
# Loop through each column in the dataframe
for column in bank_additional_full_df.columns:
    num_unique_values = bank_additional_full_df[column].nunique()

    print(f"\n----- Column: '{column}' -----")
```

```python
    print(f"Number of unique values: {num_unique_values}")

    # Set a threshold to decide whether to print all unique values
    # This avoids printing thousands of unique values for continuous columns␣
↪like 'age' or 'duration'
    if num_unique_values < 15:
        # Sort the values to make them easier to read
        unique_values = sorted(bank_additional_full_df[column].unique())
        print(f"Unique values: {unique_values}")
    else:
        # For columns with many unique values, we just note that it's a␣
↪high-cardinality feature
        # We can show a small sample of the unique values
        sample_unique_values = list(bank_additional_full_df[column].unique())[:
↪5]
        print(f"Values: [High Cardinality Feature - Sample:␣
↪{sample_unique_values}...]")
```

```
----- Column: 'age' -----
Number of unique values: 78
Values: [High Cardinality Feature - Sample: [np.int64(56), np.int64(57),
np.int64(37), np.int64(40), np.int64(45)]…]

----- Column: 'job' -----
Number of unique values: 12
Unique values: ['admin.', 'blue-collar', 'entrepreneur', 'housemaid',
'management', 'retired', 'self-employed', 'services', 'student', 'technician',
'unemployed', 'unknown']

----- Column: 'marital' -----
Number of unique values: 4
Unique values: ['divorced', 'married', 'single', 'unknown']

----- Column: 'education' -----
Number of unique values: 8
Unique values: ['basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate',
'professional.course', 'university.degree', 'unknown']

----- Column: 'default' -----
Number of unique values: 3
Unique values: ['no', 'unknown', 'yes']

----- Column: 'housing' -----
Number of unique values: 3
Unique values: ['no', 'unknown', 'yes']
```

```
----- Column: 'loan' -----
Number of unique values: 3
Unique values: ['no', 'unknown', 'yes']

----- Column: 'contact' -----
Number of unique values: 2
Unique values: ['cellular', 'telephone']

----- Column: 'month' -----
Number of unique values: 10
Unique values: ['apr', 'aug', 'dec', 'jul', 'jun', 'mar', 'may', 'nov', 'oct',
'sep']

----- Column: 'day_of_week' -----
Number of unique values: 5
Unique values: ['fri', 'mon', 'thu', 'tue', 'wed']

----- Column: 'duration' -----
Number of unique values: 1544
Values: [High Cardinality Feature - Sample: [np.int64(261), np.int64(149),
np.int64(226), np.int64(151), np.int64(307)]…]

----- Column: 'campaign' -----
Number of unique values: 42
Values: [High Cardinality Feature - Sample: [np.int64(1), np.int64(2),
np.int64(3), np.int64(4), np.int64(5)]…]

----- Column: 'pdays' -----
Number of unique values: 27
Values: [High Cardinality Feature - Sample: [np.int64(999), np.int64(6),
np.int64(4), np.int64(3), np.int64(5)]…]

----- Column: 'previous' -----
Number of unique values: 8
Unique values: [np.int64(0), np.int64(1), np.int64(2), np.int64(3), np.int64(4),
np.int64(5), np.int64(6), np.int64(7)]

----- Column: 'poutcome' -----
Number of unique values: 3
Unique values: ['failure', 'nonexistent', 'success']

----- Column: 'emp.var.rate' -----
Number of unique values: 10
Unique values: [np.float64(-3.4), np.float64(-3.0), np.float64(-2.9),
np.float64(-1.8), np.float64(-1.7), np.float64(-1.1), np.float64(-0.2),
np.float64(-0.1), np.float64(1.1), np.float64(1.4)]

----- Column: 'cons.price.idx' -----
```

```
Number of unique values: 26
Values: [High Cardinality Feature - Sample: [np.float64(93.994),
np.float64(94.465), np.float64(93.918), np.float64(93.444),
np.float64(93.798)]…]

----- Column: 'cons.conf.idx' -----
Number of unique values: 26
Values: [High Cardinality Feature - Sample: [np.float64(-36.4),
np.float64(-41.8), np.float64(-42.7), np.float64(-36.1), np.float64(-40.4)]…]

----- Column: 'euribor3m' -----
Number of unique values: 316
Values: [High Cardinality Feature - Sample: [np.float64(4.857),
np.float64(4.856), np.float64(4.855), np.float64(4.859), np.float64(4.86)]…]

----- Column: 'nr.employed' -----
Number of unique values: 11
Unique values: [np.float64(4963.6), np.float64(4991.6), np.float64(5008.7),
np.float64(5017.5), np.float64(5023.5), np.float64(5076.2), np.float64(5099.1),
np.float64(5176.3), np.float64(5191.0), np.float64(5195.8), np.float64(5228.1)]

----- Column: 'y' -----
Number of unique values: 2
Unique values: ['no', 'yes']
```

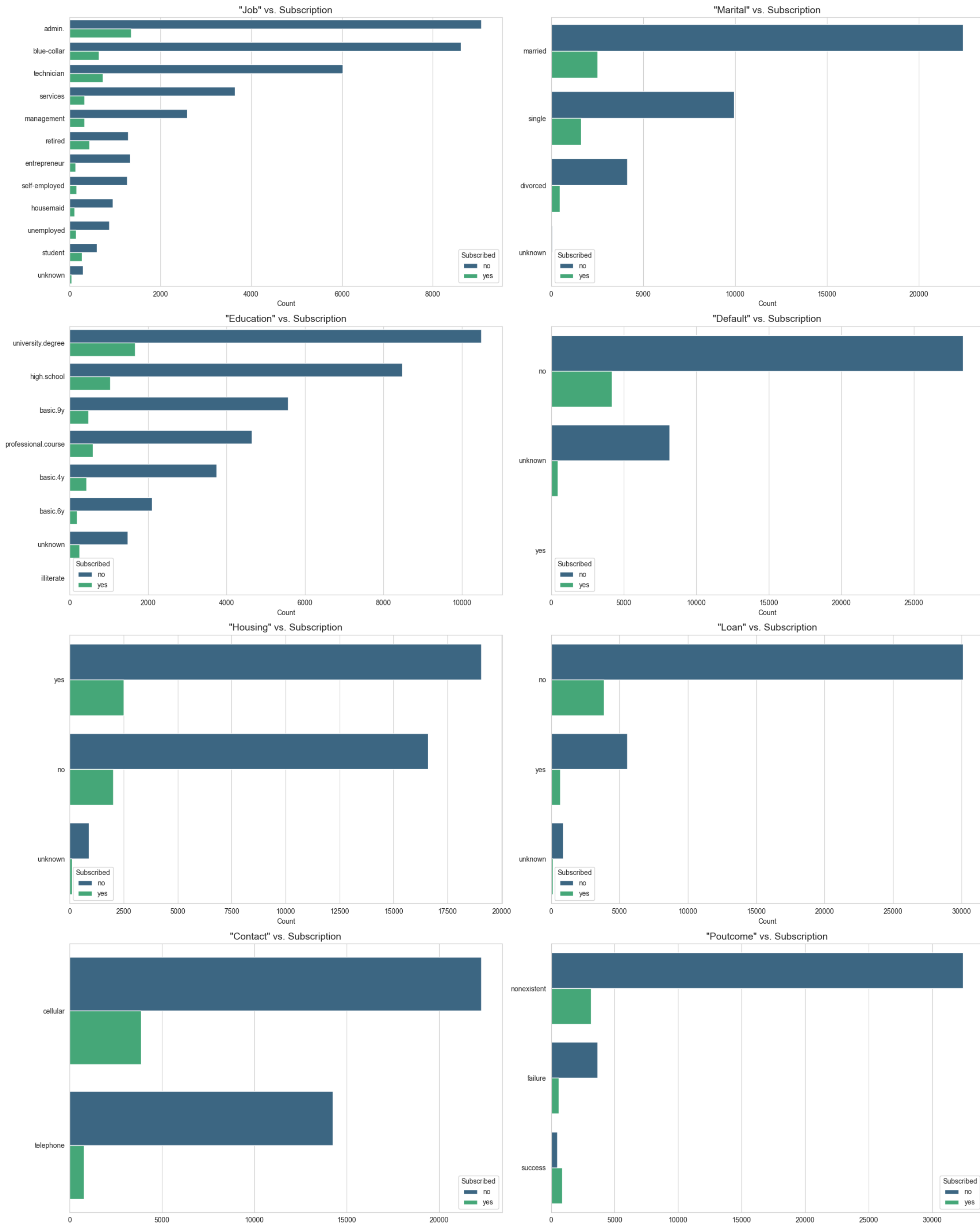
1.4 1.4. Feature Analysis: Feature vs. Target

Now we'll analyze how each feature relates to the subscription outcome y. This will help us identify potentially predictive features.

1.4.1 1.4.1 Categorical Features vs. Target ('y')

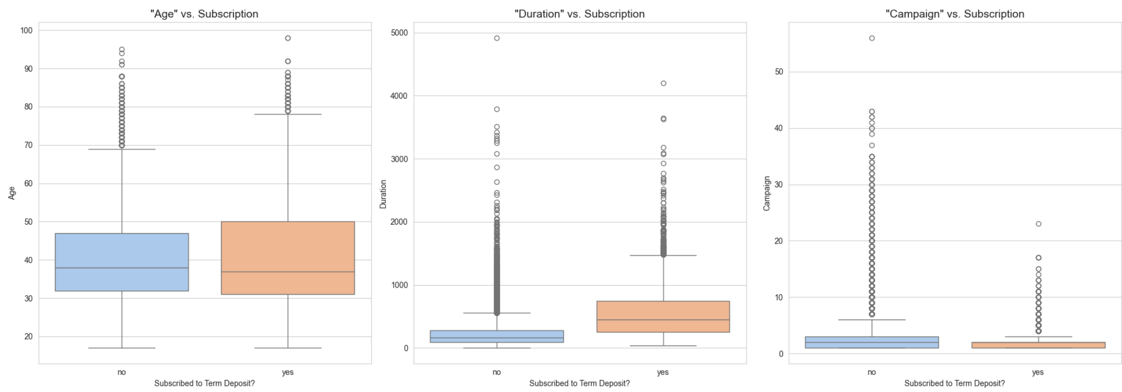
[10]:
```python
# List of key categorical features to analyze
cat_features_to_plot = ['job', 'marital', 'education', 'default', 'housing',␣
↪'loan', 'contact', 'poutcome']

# Create plots
fig, axes = plt.subplots(4, 2, figsize=(20, 25))
axes = axes.flatten()

for i, col in enumerate(cat_features_to_plot):
    # Use hue to show the distribution of the target variable for each category
    sns.countplot(y=col, data=bank_additional_full_df, ax=axes[i],␣
↪order=bank_additional_full_df[col].value_counts().index, hue='y',␣
↪palette='viridis')
    axes[i].set_title(f'"{col.capitalize()}" vs. Subscription', fontsize=14)
    axes[i].set_xlabel('Count')
    axes[i].set_ylabel('')
```

```
    axes[i].legend(title='Subscribed')

plt.tight_layout()
plt.show()
```



1.4.2 1.4.2 Numerical Features vs. Target ('y')

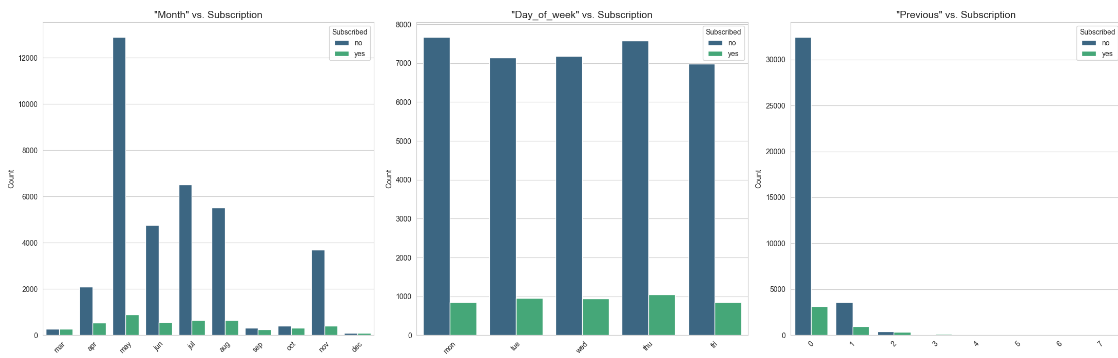
[11]:
```
# List of key numerical features to analyze
num_features_to_plot = ['age', 'duration', 'campaign']

# Create boxplots
fig, axes = plt.subplots(1, 3, figsize=(20, 7))

for i, col in enumerate(num_features_to_plot):
    sns.boxplot(x='y', y=col, data=bank_additional_full_df, ax=axes[i],hue="y",␣
 ↪palette='pastel', legend=False)
    axes[i].set_title(f'"{col.capitalize()}" vs. Subscription', fontsize=14)
    axes[i].set_xlabel('Subscribed to Term Deposit?')
    axes[i].set_ylabel(col.capitalize())

plt.tight_layout()
plt.show()
```



1.4.3 1.4.3 Time & Previous Campaign Features vs. Target ('y')

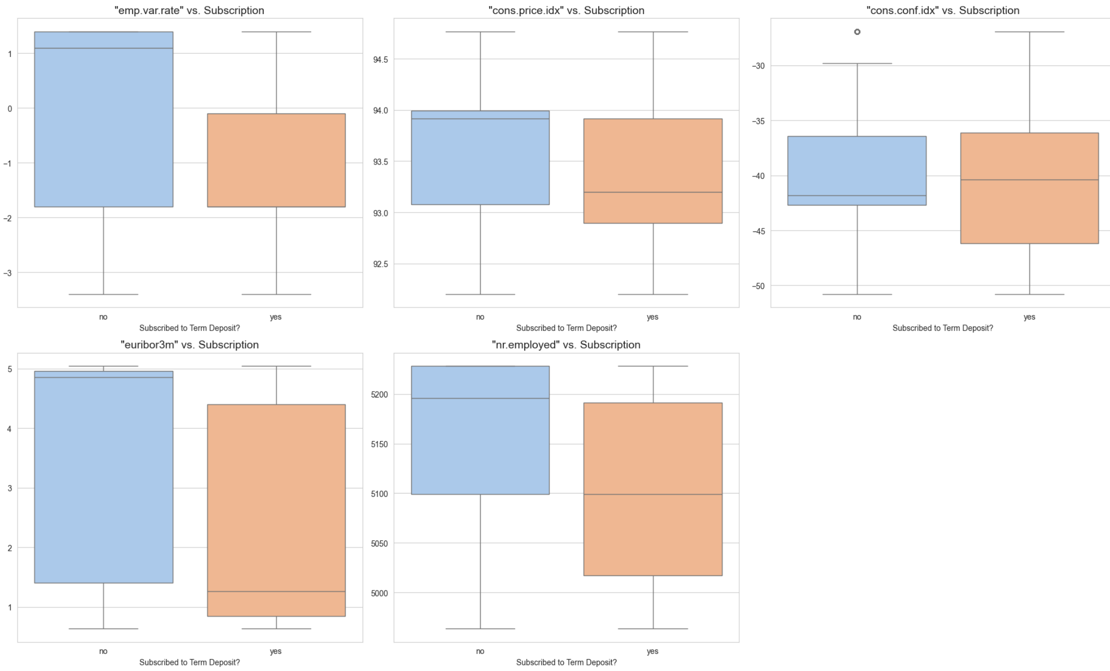
[12]:
```
# Features to analyze
features_to_plot = ['month', 'day_of_week', 'previous']

# Define a chronological order for months and days
month_order = ['mar', 'apr', 'may', 'jun', 'jul', 'aug', 'sep', 'oct', 'nov',␣
 ↪'dec']
day_order = ['mon', 'tue', 'wed', 'thu', 'fri']
order_map = {'month': month_order, 'day_of_week': day_order, 'previous':␣
 ↪sorted(bank_additional_full_df['previous'].unique())}


fig, axes = plt.subplots(1, 3, figsize=(22, 7))
axes = axes.flatten()
```

```python
for i, col in enumerate(features_to_plot):
    sns.countplot(x=col, data=bank_additional_full_df, ax=axes[i], order=order_map[col], hue='y', palette='viridis')
    axes[i].set_title(f'"{col.capitalize()}" vs. Subscription', fontsize=14)
    axes[i].set_ylabel('Count')
    axes[i].set_xlabel('')
    axes[i].tick_params(axis='x', rotation=45) # Rotate labels for readability
    axes[i].legend(title='Subscribed')

plt.tight_layout()
plt.show()
```



1.4.4 1.4.4 Social & Economic Features vs. Target ('y')

[13]:
```python
# List of social and economic features
social_econ_features = ['emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed']

# Create boxplots
fig, axes = plt.subplots(2, 3, figsize=(20, 12))
axes = axes.flatten()
for i, col in enumerate(social_econ_features):
    sns.boxplot(x='y', y=col, data=bank_additional_full_df, ax=axes[i], palette='pastel', hue='y', legend=False)
    axes[i].set_title(f'"{col}" vs. Subscription', fontsize=14)
    axes[i].set_xlabel('Subscribed to Term Deposit?')
    axes[i].set_ylabel('')
# Hide the empty subplot
fig.delaxes(axes[5])
plt.tight_layout()
plt.show()
```



1.4.5 1.4.5 Visual Feature Analysis Summary

A. Categorical Features vs. Target ('y')

- **Job & Education**: Administrative staff, technicians, and blue-collar workers form the largest groups of clients contacted. However, students and retired individuals show a proportionally higher subscription rate ("yes") compared to other job categories. Clients with a university degree were contacted most frequently, but subscription rates appear relatively consistent across different education levels, with a slight increase for those with higher education.
- **Marital & Default Status**: Married individuals are the largest client segment, followed by singles. The "Default" status is overwhelmingly "no," and very few clients with a "yes" status subscribed. This suggests that clients with a history of credit default are unlikely to subscribe.
- **Housing & Loan**: The subscription rate is higher for clients who do not have an existing housing loan. Similarly, clients without a personal loan are more likely to subscribe than those with one.
- **Contact & Previous Campaign Outcome (Poutcome)**: Contacting clients via "cellular" is associated with a much higher subscription rate than "telephone". As expected, a "success" outcome from a previous campaign is a strong predictor of a "yes" for the current campaign.

B. Numerical Features vs. Target ('y')

- **Age**: The age distribution for both subscribers and non-subscribers is similar, with the median age for subscribers appearing slightly higher than for non-subscribers.
- **Duration**: The duration of the last contact is significantly higher for clients who subscribed. This is a key insight but must be handled with care, as call duration is not known until after the call is made. Thus, it cannot be used as a predictive feature for a pre-call model.

- **Campaign**: The number of contacts during the campaign is heavily skewed towards the lower end for both groups. However, the median number of contacts for subscribers is slightly lower than for non-subscribers, suggesting that fewer contacts are often more effective.

C. Time & Previous Campaign Features vs. Target ('y')

- **Month & Day of the Week**: Subscription rates vary significantly by month, with the highest success rates appearing in March, September, October, and December. The day of the week does not show a significant variation in subscription rates.
- **Previous**: A higher number of previous contacts (before the current campaign) is associated with a higher likelihood of subscribing, although the vast majority of clients had no previous contact.

D. Social & Economic Features vs. Target ('y')

- The boxplots for social and economic indicators show clear distinctions between subscribers and non-subscribers:
  - **Subscribers ("yes") are associated with**:
    * Lower (more negative) employment variation rates (`emp.var.rate`).
    * Lower consumer price indexes (`cons.price.idx`).
    * Higher (less negative) consumer confidence indexes (`cons.conf.idx`).
    * Lower 3-month Euribor rates (`euribor3m`).
    * Lower numbers of employees (`nr.employed`).
- These trends suggest that clients are more likely to subscribe during periods of lower economic pressure (e.g., lower interest rates, lower employment figures, and higher consumer confidence).

1.5 1.5 Quantifying Feature Relationships

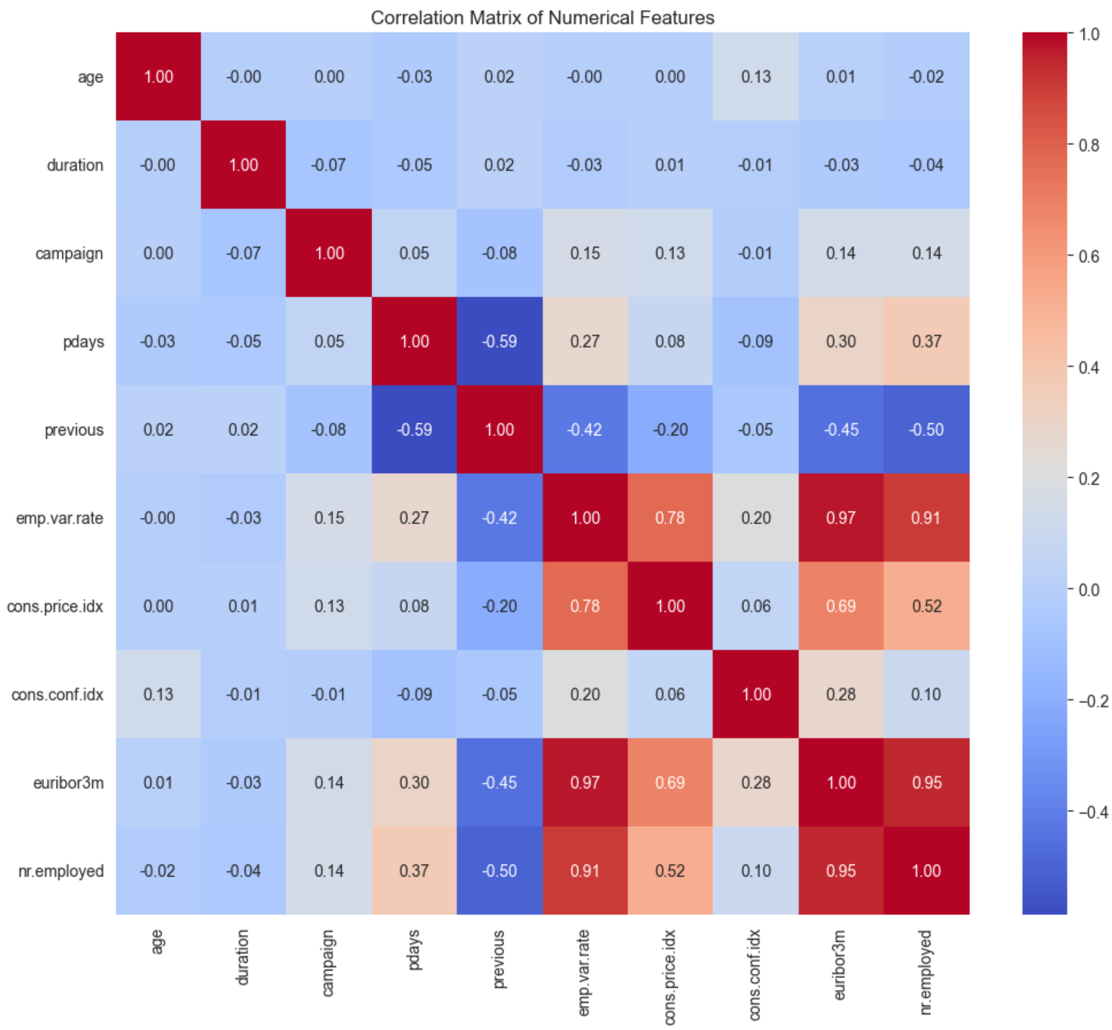
The next logical step in the Exploratory Data Analysis (EDA) would be to quantify the relationships observed visually in the last section. The plots have shown a good intuition about the data, and the next step is to generate concrete numbers to support these findings before moving to preprocessing the data.

1.5.1 1.5.1 Correlation of numerical Columns

On how to interprete this correlation materix refer to this website which explains how to read a correlation matrix.

[16]:
```python
# Select only numerical columns for correlation
numerical_cols = bank_additional_full_df.select_dtypes(include=np.number).
 ↪columns

# Calculate the correlation matrix
corr_matrix = bank_additional_full_df[numerical_cols].corr()

# Plot the heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Matrix of Numerical Features')
```

```
plt.show()
```

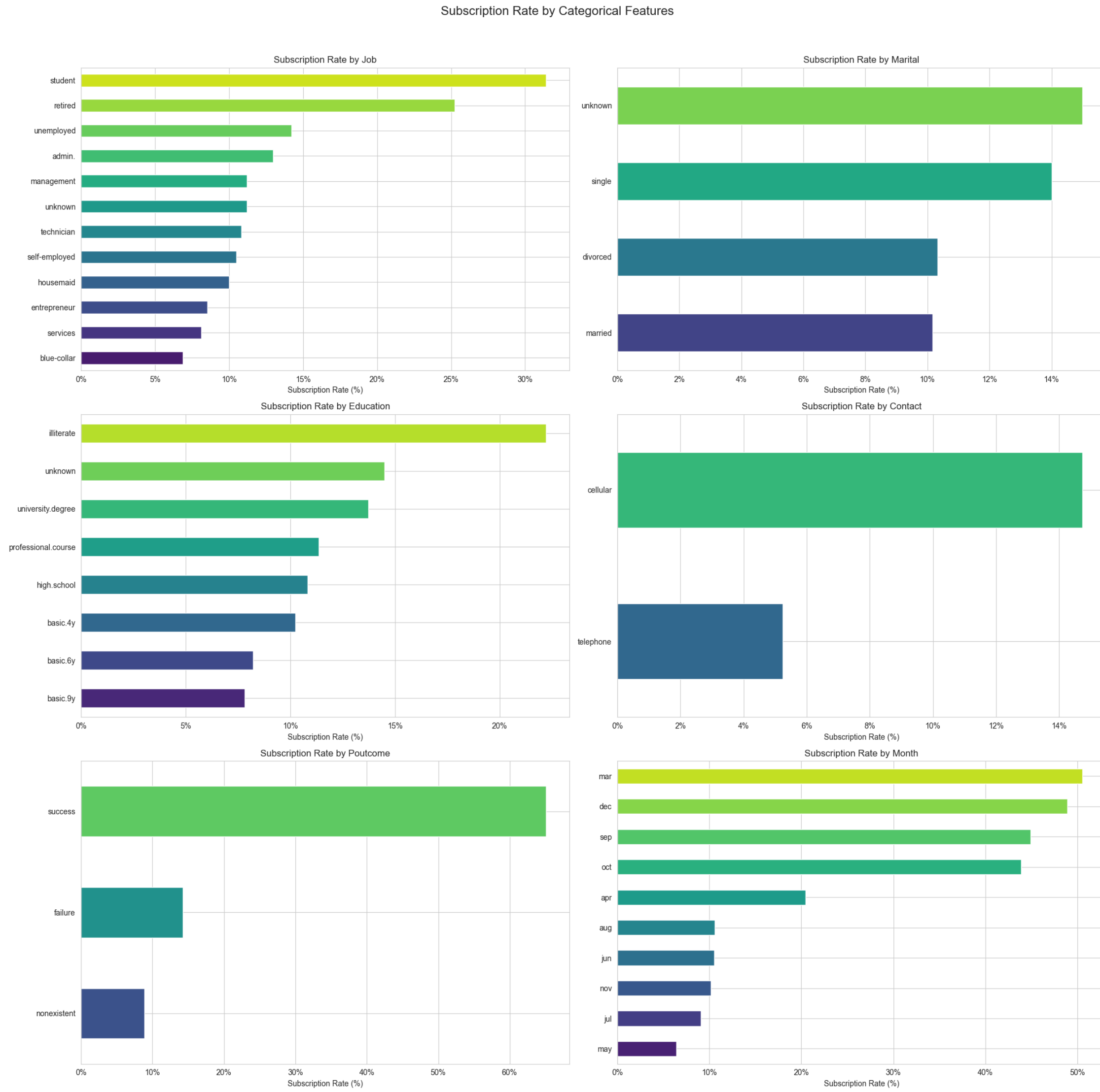


1.5.2 1.5.2 Subscription Rate Analysis

[21]:
```
# First, create a copy and convert 'y' to a numerical format 1: yes 0: no
df_rate = bank_additional_full_df.copy()
df_rate['y_numeric'] = df_rate['y'].apply(lambda x: 1 if x == 'yes' else 0)
```

[22]:
```
# We will reuse the df_rate DataFrame with the 'y_numeric' column
cat_features = ['job', 'marital', 'education', 'contact', 'poutcome', 'month']

fig, axes = plt.subplots(3, 2, figsize=(20, 20))
axes = axes.flatten()
fig.suptitle('Subscription Rate by Categorical Features', fontsize=16)
```

```
for i, col in enumerate(cat_features):
    # Calculate subscription rate
    rate = df_rate.groupby(col)['y_numeric'].mean().sort_values(ascending=True)

    # Plot
    rate.plot(kind='barh', ax=axes[i], color=sns.color_palette('viridis',␣
 ↪len(rate)))
    axes[i].set_title(f'Subscription Rate by {col.capitalize()}')
    axes[i].set_xlabel('Subscription Rate (%)')
    axes[i].set_ylabel('')
    # Format x-axis as percentage
    axes[i].xaxis.set_major_formatter(plt.FuncFormatter(lambda x, _: '{:.0%}'.
 ↪format(x)))

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```



1.5.3 Correlation Analysis Summary

Correlation Matrix of Numerical Features The correlation matrix reveals strong relationships between several of the socio-economic indicator variables. - High Multicollinearity: There are very strong positive correlations between `emp.var.rate, euribor3m, and nr.employed`, with correlation coefficients ranging from 0.91 to 0.97. This indicates that these features measure similar underlying economic conditions. For certain models, like logistic regression, this multicollinearity can be problematic, and we may consider removing some of these redundant features during the feature selection phase. - Other Correlations: A moderate negative correlation exists between previous (number of contacts before this campaign) and pdays (days since last contact), with a coefficient of -0.59. This makes sense, as clients with more previous contacts are likely to have been contacted more recently.

Subscription Rate Analysis Calculating the exact subscription rates confirms the visual insights from the earlier plots and provides precise metrics.

- By Job: Students (31.4%) and retired clients (25.2%) have the highest likelihood of subscribing to a term deposit. In contrast, blue-collar workers (6.9%) have the lowest subscription rate. This reinforces that targeting specific client professions could significantly increase campaign efficiency.
- By Previous Outcome: The outcome of a previous campaign is an extremely powerful indicator. Clients with a "success" in a prior campaign have a 65.1% subscription rate, which is dramatically higher than for clients with no previous contact ("nonexistent" at 8.8%) or a previous "failure" (14.2%).

1.6 1.6 Conclusion of Exploratory Data Analysis (EDA)

This EDA has identified key patterns and characteristics within the dataset that will be needed during our modeling strategy.

1. Strong Predictors Identified: Several features show a strong relationship with the client's decision to subscribe. The most influential appear to be the outcome of the previous campaign (`poutcome`), the month of contact, the contact method (`contact`), and the socio-economic indicators (`emp.var.rate, euribor3m, etc.`). Client job type also shows significant variance in subscription rates.

2. Data Leakage for Model: The duration feature is highly correlated with the outcome but must be excluded from the predictive model, since this information is not available before an actual call is made.

3. Data Quality & Preprocessing Needs:

- "Unknown" Values: Several key categorical features contain "unknown" entries, which will need to be handled, either by treating them as a distinct category, imputing them with different categories or replacing as NAN values.
- Class Imbalance: The target variable 'y' is highly imbalanced, with a large majority of non-subscribers. This must be addressed during the modeling phase to prevent model bias.
- Multicollinearity: The high correlation among socio-economic features suggests that feature selection is an important step.
  - When features are highly correlated, it becomes difficult to distinguish their individual effects on the target variable.
  - Multicollinearity can be problematic for certain models, such as logistic regression
  - Highly correlated features provide redundant information, thus by removing them we can reduce model complexity

2 2 Data Preprocessing

Following the EDA, the next phase is Data Preprocessing. While the EDA serves as a diagnostic investigation, providing a deep understanding of the dataset's structure, underlying patterns, and most importantly, its limitations and potential issues.

The analysis revealed mentioned challenges that must be addressed before modeling such as: - the presence of "unknown" values in categorical features and the encoding of categorical features -

significant class imbalance in the target variable - high multicollinearity among the socio-economic indicators - data leakage risk from the duration feature

[95]:
```
df = bank_additional_full_df.copy()
df
```

[95]:
```
       age          job  marital            education  default housing loan  \
0       56    housemaid  married             basic.4y       no      no   no
1       57     services  married          high.school  unknown      no   no
2       37     services  married          high.school       no     yes   no
3       40       admin.  married             basic.6y       no      no   no
4       56     services  married          high.school       no      no  yes
…      …          …        …                  …      …      …    …
41183   73      retired  married  professional.course       no     yes   no
41184   46  blue-collar  married  professional.course       no      no   no
41185   56      retired  married    university.degree       no     yes   no
41186   44   technician  married  professional.course       no      no   no
41187   74      retired  married  professional.course       no     yes   no

         contact month day_of_week  …  campaign  pdays  previous  \
0      telephone   may         mon  …         1    999         0
1      telephone   may         mon  …         1    999         0
2      telephone   may         mon  …         1    999         0
3      telephone   may         mon  …         1    999         0
4      telephone   may         mon  …         1    999         0
…            …   …           …  …       …      …         …
41183   cellular   nov         fri  …         1    999         0
41184   cellular   nov         fri  …         1    999         0
41185   cellular   nov         fri  …         2    999         0
41186   cellular   nov         fri  …         1    999         0
41187   cellular   nov         fri  …         3    999         1

          poutcome emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  \
0      nonexistent          1.1          93.994          -36.4      4.857
1      nonexistent          1.1          93.994          -36.4      4.857
2      nonexistent          1.1          93.994          -36.4      4.857
3      nonexistent          1.1          93.994          -36.4      4.857
4      nonexistent          1.1          93.994          -36.4      4.857
…              …          …             …            …        …
41183  nonexistent         -1.1          94.767          -50.8      1.028
41184  nonexistent         -1.1          94.767          -50.8      1.028
41185  nonexistent         -1.1          94.767          -50.8      1.028
41186  nonexistent         -1.1          94.767          -50.8      1.028
41187      failure         -1.1          94.767          -50.8      1.028

       nr.employed   y
0           5191.0  no
```

```
1           5191.0   no
2           5191.0   no
3           5191.0   no
4           5191.0   no
…             …  …
41183       4963.6  yes
41184       4963.6   no
41185       4963.6   no
41186       4963.6  yes
41187       4963.6   no

[41188 rows x 21 columns]
```

2.1 2.1 Address Data Leakage by Removing 'duration' column

[96]:
```
# Remove the 'duration' column to prevent data leakage
df.drop('duration', axis=1, inplace=True)
```

2.2 2.2 Encode the Target Variable 'y'

Machine learning models require numerical inputs, so our first and simplest step is to convert the y column's values from "yes" and "no" to a binary format (1 and 0).

[97]:
```
df['y'] = df['y'].apply(lambda x: 1 if x == 'yes' else 0)
df
```

[97]:
```
       age          job  marital            education  default housing loan  \
0       56    housemaid  married             basic.4y       no      no   no
1       57     services  married          high.school  unknown      no   no
2       37     services  married          high.school       no     yes   no
3       40       admin.  married             basic.6y       no      no   no
4       56     services  married          high.school       no      no  yes
…     …           …       …                   …      …       …    …
41183   73      retired  married  professional.course       no     yes   no
41184   46  blue-collar  married  professional.course       no      no   no
41185   56      retired  married    university.degree       no     yes   no
41186   44   technician  married  professional.course       no      no   no
41187   74      retired  married  professional.course       no     yes   no

         contact month day_of_week  campaign  pdays  previous     poutcome  \
0      telephone   may         mon         1    999         0  nonexistent
1      telephone   may         mon         1    999         0  nonexistent
2      telephone   may         mon         1    999         0  nonexistent
3      telephone   may         mon         1    999         0  nonexistent
4      telephone   may         mon         1    999         0  nonexistent
…          …   …         …       …       …       …          …
41183   cellular   nov         fri         1    999         0  nonexistent
```

```
41184   cellular    nov         fri          1     999          0  nonexistent
41185   cellular    nov         fri          2     999          0  nonexistent
41186   cellular    nov         fri          1     999          0  nonexistent
41187   cellular    nov         fri          3     999          1      failure

       emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  nr.employed  y
0               1.1          93.994          -36.4      4.857       5191.0  0
1               1.1          93.994          -36.4      4.857       5191.0  0
2               1.1          93.994          -36.4      4.857       5191.0  0
3               1.1          93.994          -36.4      4.857       5191.0  0
4               1.1          93.994          -36.4      4.857       5191.0  0
…               …               …              …         …           … ..
41183          -1.1          94.767          -50.8      1.028       4963.6  1
41184          -1.1          94.767          -50.8      1.028       4963.6  0
41185          -1.1          94.767          -50.8      1.028       4963.6  0
41186          -1.1          94.767          -50.8      1.028       4963.6  1
41187          -1.1          94.767          -50.8      1.028       4963.6  0

[41188 rows x 20 columns]
```

Why do we need to encode categoriacal columns? Machine learning algorithms are based on mathematical equations and can only process numerical data. They cannot understand text labels like 'yes' or 'no' directly. In this case we the y category with 1 and 0 since it has only 2 options. **But what about features that have more categories?**

2.3 2.3 Encoding categoriacal features with more than 2 categories using One-Hot-Encoding

The principle of one-hot encoding works exactly the same way, whether a feature has two categories or many more. It simply expands to create a new binary column for every unique category.

So let's encode all categorical columns.

[98]:
```python
# 1. Identify all categorical columns that need encoding
# These are the columns with string values (object dtype)
categorical_features = df.select_dtypes(include=['object']).columns
categorical_features
```

[98]:
```
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact',
       'month', 'day_of_week', 'poutcome'],
      dtype='object')
```

[99]:
```python
# 2. Apply one-hot encoding using pd.get_dummies()
df_encoded = pd.get_dummies(df, columns=categorical_features,␣
 ↪drop_first=False,dtype=int)

# 3. Display the results
```

```python
print(f"Original shape of the DataFrame: {df.shape}")
print(f"Shape after one-hot encoding: {df_encoded.shape}\n")
```

```
Original shape of the DataFrame: (41188, 20)
Shape after one-hot encoding: (41188, 63)
```

[100]:
```python
df_encoded.head(5)
```

[100]:
```
   age  campaign  pdays  previous  emp.var.rate  cons.price.idx  \
0   56         1    999         0           1.1          93.994
1   57         1    999         0           1.1          93.994
2   37         1    999         0           1.1          93.994
3   40         1    999         0           1.1          93.994
4   56         1    999         0           1.1          93.994

   cons.conf.idx  euribor3m  nr.employed  y  …  month_oct  month_sep  \
0          -36.4      4.857       5191.0  0  …          0          0
1          -36.4      4.857       5191.0  0  …          0          0
2          -36.4      4.857       5191.0  0  …          0          0
3          -36.4      4.857       5191.0  0  …          0          0
4          -36.4      4.857       5191.0  0  …          0          0

   day_of_week_fri  day_of_week_mon  day_of_week_thu  day_of_week_tue  \
0                0                1                0                0
1                0                1                0                0
2                0                1                0                0
3                0                1                0                0
4                0                1                0                0

   day_of_week_wed  poutcome_failure  poutcome_nonexistent  poutcome_success
0                0                 0                     1                 0
1                0                 0                     1                 0
2                0                 0                     1                 0
3                0                 0                     1                 0
4                0                 0                     1                 0

[5 rows x 63 columns]
```

2.4 2.4 Separating Feature and Target Variable

[101]:
```python
# Separate the features (X) from the target variable (y)
X = df_encoded.drop('y', axis=1)
y = df_encoded['y']

print(f"Shape of features (X): {X.shape}")
print(f"Shape of target (y): {y.shape}")
```

```
Shape of features (X): (41188, 62)
Shape of target (y): (41188,)
```

2.5 2.5 Split Data into Training and Testing Sets

[102]:
```
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets (80% train, 20% test)
# 'stratify=y' ensures that the proportion of subscribers is the same in both␣
↪sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42, stratify=y)

print(f"Training set size: {X_train.shape[0]} samples")
print(f"Testing set size: {X_test.shape[0]} samples")
```

```
Training set size: 32950 samples
Testing set size: 8238 samples
```

3 3. Creating a Baseline Model using LogisticRegression for this Classification Task

[103]:
```
from sklearn.linear_model import LogisticRegression
# 1. Initialize the Logistic Regression model
# We'll set max_iter to a higher value to ensure the model's algorithm␣
↪converges.
log_reg_baseline = LogisticRegression(random_state=42, max_iter=500)
```

[104]:
```
# 2. Train the model on the (unscaled, imbalanced) training data
print("Training the baseline logistic regression model...")
log_reg_baseline.fit(X_train, y_train)
print("Model training complete.")
```

```
Training the baseline logistic regression model…
```

```
/Users/tomyle/Pycharm/xai_capstone/.venv/lib/python3.12/site-
packages/sklearn/linear_model/_linear_loss.py:200: RuntimeWarning: divide by
zero encountered in matmul
  raw_prediction = X @ weights + intercept
/Users/tomyle/Pycharm/xai_capstone/.venv/lib/python3.12/site-
packages/sklearn/linear_model/_linear_loss.py:200: RuntimeWarning: overflow
encountered in matmul
  raw_prediction = X @ weights + intercept
/Users/tomyle/Pycharm/xai_capstone/.venv/lib/python3.12/site-
packages/sklearn/linear_model/_linear_loss.py:200: RuntimeWarning: invalid value
encountered in matmul
  raw_prediction = X @ weights + intercept
/Users/tomyle/Pycharm/xai_capstone/.venv/lib/python3.12/site-
```

```
packages/sklearn/linear_model/_linear_loss.py:330: RuntimeWarning: divide by
zero encountered in matmul
  grad[:n_features] = X.T @ grad_pointwise + l2_reg_strength * weights
/Users/tomyle/Pycharm/xai_capstone/.venv/lib/python3.12/site-
packages/sklearn/linear_model/_linear_loss.py:330: RuntimeWarning: overflow
encountered in matmul
  grad[:n_features] = X.T @ grad_pointwise + l2_reg_strength * weights
/Users/tomyle/Pycharm/xai_capstone/.venv/lib/python3.12/site-
packages/sklearn/linear_model/_linear_loss.py:330: RuntimeWarning: invalid value
encountered in matmul
  grad[:n_features] = X.T @ grad_pointwise + l2_reg_strength * weights
```

```
Model training complete.
```

```
/Users/tomyle/Pycharm/xai_capstone/.venv/lib/python3.12/site-
packages/sklearn/linear_model/_logistic.py:470: ConvergenceWarning: lbfgs failed
to converge after 500 iteration(s) (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=500).
You might also want to scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

[105]:

```python
# 3. Make predictions on the test data
y_pred_baseline = log_reg_baseline.predict(X_test)
print("\nPredictions have been made on the test set.")
```

```
Predictions have been made on the test set.
```

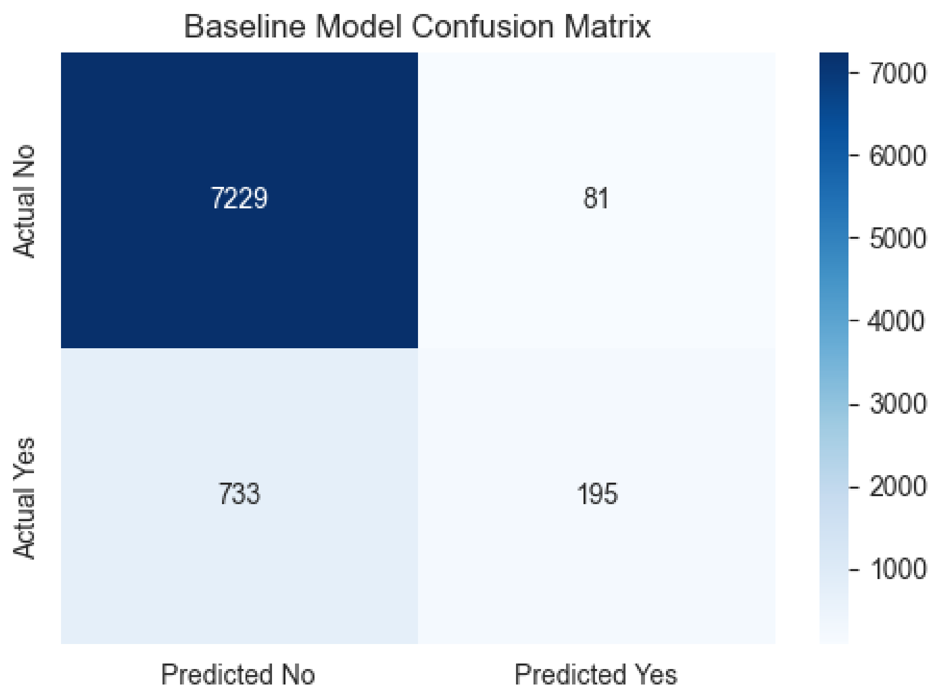
```
/Users/tomyle/Pycharm/xai_capstone/.venv/lib/python3.12/site-
packages/sklearn/utils/extmath.py:203: RuntimeWarning: divide by zero
encountered in matmul
  ret = a @ b
/Users/tomyle/Pycharm/xai_capstone/.venv/lib/python3.12/site-
packages/sklearn/utils/extmath.py:203: RuntimeWarning: overflow encountered in
matmul
  ret = a @ b
/Users/tomyle/Pycharm/xai_capstone/.venv/lib/python3.12/site-
packages/sklearn/utils/extmath.py:203: RuntimeWarning: invalid value encountered
in matmul
  ret = a @ b
```

3.1 3. 1 Model Evalutation

3.1.1 3.1.1 Confusion Matrix

[106]:
```
from sklearn.metrics import classification_report, confusion_matrix
# --- 1. Confusion Matrix ---
print("--- Confusion Matrix ---")
cm = confusion_matrix(y_test, y_pred_baseline)

# For a nicer plot
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=['Predicted No', 'Predicted Yes'],
            yticklabels=['Actual No', 'Actual Yes'])
plt.title('Baseline Model Confusion Matrix')
plt.show()

# --- 2. Classification Report ---
print("\n--- Classification Report ---")
report = classification_report(y_test, y_pred_baseline, target_names=['No␣
↪(Class 0)', 'Yes (Class 1)'])
print(report)

TN, FP, FN, TP = cm.ravel()
# 2. Calculate the metrics using their formulas
# Accuracy: (TP + TN) / Total
accuracy = (TP + TN) / (TP + TN + FP + FN)
# Precision: TP / (TP + FP)
# How many of the predicted positives were actually positive?
precision = TP / (TP + FP)
# Sensitivity (Recall or True Positive Rate): TP / (TP + FN)
# How many of the actual positives were correctly identified?
sensitivity = TP / (TP + FN)
# Specificity (True Negative Rate): TN / (TN + FP)
# How many of the actual negatives were correctly identified?
specificity = TN / (TN + FP)


# 3. Print the results in a clear format
print("--- Detailed Metrics ---")
print(f"Accuracy:    {accuracy:.4f}  (Overall correctness)")
print(f"Precision:   {precision:.4f}  (Correctness of positive predictions)")
print(f"Sensitivity: {sensitivity:.4f}  (Ability to find all positive samples -␣
↪a.k.a. Recall)")
print(f"Specificity: {specificity:.4f}  (Ability to find all negative samples)")
```

```
--- Confusion Matrix ---
```




```
--- Classification Report ---
               precision    recall  f1-score   support

 No (Class 0)       0.91      0.99      0.95      7310
Yes (Class 1)       0.71      0.21      0.32       928

     accuracy                           0.90      8238
    macro avg       0.81      0.60      0.64      8238
 weighted avg       0.89      0.90      0.88      8238

--- Detailed Metrics ---
Accuracy:    0.9012  (Overall correctness)
Precision:   0.7065  (Correctness of positive predictions)
Sensitivity: 0.2101  (Ability to find all positive samples - a.k.a. Recall)
Specificity: 0.9889  (Ability to find all negative samples)
```

The Goal is to improve the TP and TN

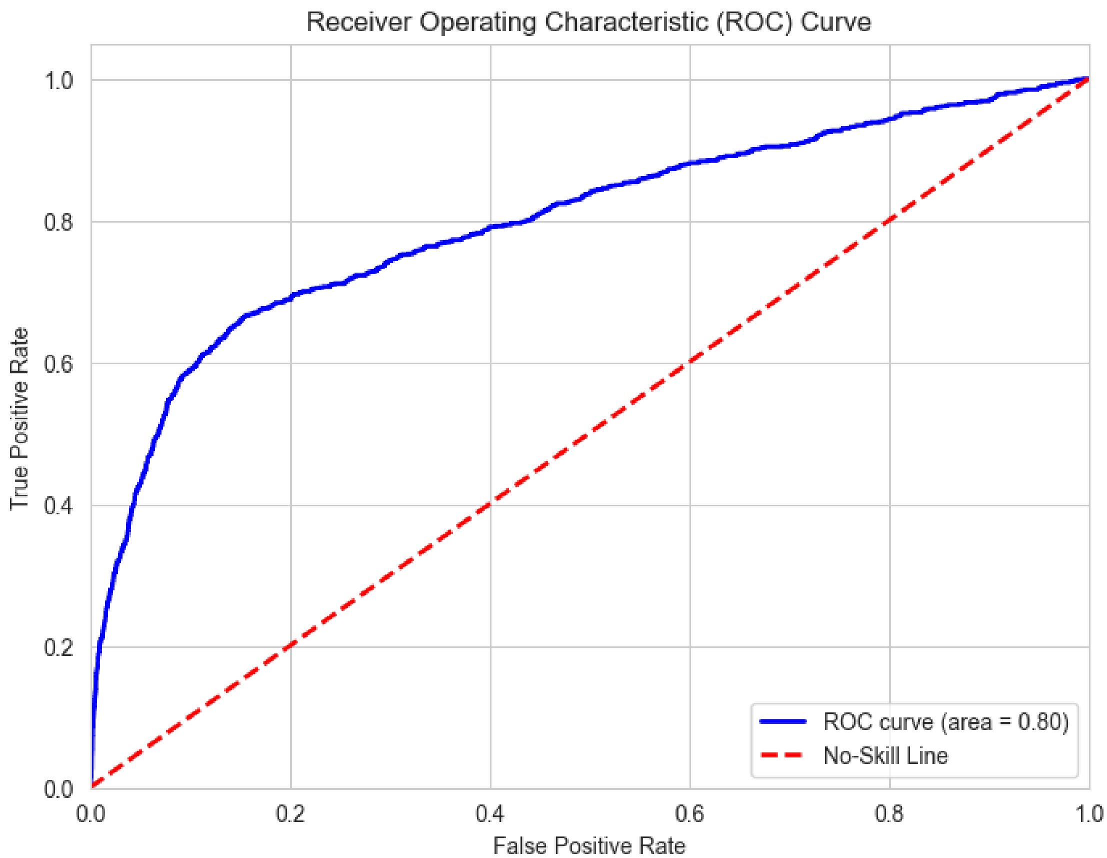
3.1.2 3.1.2 Roc Curve and ROCAUC

[107]:
```python
from sklearn.metrics import roc_curve, roc_auc_score
# 1. Get prediction probabilities for the positive class (Class 1)
y_pred_probs = log_reg_baseline.predict_proba(X_test)[:, 1]
```

```python
# 2. Calculate the ROC curve points
fpr, tpr, thresholds = roc_curve(y_test, y_pred_probs)

# 3. Calculate the ROC AUC score
roc_auc = roc_auc_score(y_test, y_pred_probs)
print(f"ROC AUC Score: {roc_auc:.4f}")

# 4. Plot the ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', lw=2, label=f'ROC curve (area = {roc_auc:.
 ↪2f})')
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--', label='No-Skill␣
 ↪Line')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```

```
ROC AUC Score: 0.7967
```

```
/Users/tomyle/Pycharm/xai_capstone/.venv/lib/python3.12/site-
packages/sklearn/utils/extmath.py:203: RuntimeWarning: divide by zero
encountered in matmul
  ret = a @ b
/Users/tomyle/Pycharm/xai_capstone/.venv/lib/python3.12/site-
packages/sklearn/utils/extmath.py:203: RuntimeWarning: overflow encountered in
matmul
  ret = a @ b
/Users/tomyle/Pycharm/xai_capstone/.venv/lib/python3.12/site-
packages/sklearn/utils/extmath.py:203: RuntimeWarning: invalid value encountered
in matmul
  ret = a @ b
```



4 Explainable AI (XAI)

4.1 Global Explainability

4.2 Local Explainablity

[ ]:

[ ]: