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

In [5]:
```
import pandas as pd
df = pd.read_csv('D:/prodigy/task3 tree/bank-additional.csv', sep=';')
```

In [6]:
```
print(df.head())
print(df.columns)
print(df.isnull().sum())
print(df.describe())
```

```
   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

     contact month day_of_week  ...  campaign  pdays  previous     poutcome  \
0   cellular   may         fri  ...         2    999         0  nonexistent
1  telephone   may         fri  ...         4    999         0  nonexistent
2  telephone   jun         wed  ...         1    999         0  nonexistent
3  telephone   jun         fri  ...         3    999         0  nonexistent
4   cellular   nov         mon  ...         1    999         0  nonexistent

  emp.var.rate  cons.price.idx  cons.conf.idx  euribor3m  nr.employed   y
0         -1.8          92.893          -46.2      1.313       5099.1  no
1          1.1          93.994          -36.4      4.855       5191.0  no
2          1.4          94.465          -41.8      4.962       5228.1  no
3          1.4          94.465          -41.8      4.959       5228.1  no
4         -0.1          93.200          -42.0      4.191       5195.8  no

[5 rows x 21 columns]
Index(['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'],
      dtype='object')
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
dtype: int64
               age     duration     campaign        pdays     previous  \
count  4119.000000  4119.000000  4119.000000  4119.000000  4119.000000
mean     40.113620   256.788055     2.537266   960.422190     0.190337
std      10.313362   254.703736     2.568159   191.922786     0.541788
min      18.000000     0.000000     1.000000     0.000000     0.000000
25%      32.000000   103.000000     1.000000   999.000000     0.000000
50%      38.000000   181.000000     2.000000   999.000000     0.000000
```

```
75%       47.000000   317.000000     3.000000   999.000000     0.000000
max       88.000000  3643.000000    35.000000   999.000000     6.000000

       emp.var.rate  cons.price.idx  cons.conf.idx    euribor3m  nr.employed
count   4119.000000     4119.000000    4119.000000  4119.000000  4119.000000
mean       0.084972       93.579704     -40.499102     3.621356  5166.481695
std        1.563114        0.579349       4.594578     1.733591    73.667904
min       -3.400000       92.201000     -50.800000     0.635000  4963.600000
25%       -1.800000       93.075000     -42.700000     1.334000  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
```

In [7]:
```
df.dropna(inplace=True)
df_encoded = pd.get_dummies(df, columns=['job', 'marital', 'education', 'default',
X = df_encoded.drop('y', axis=1)
y = df_encoded['y']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [8]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)
```

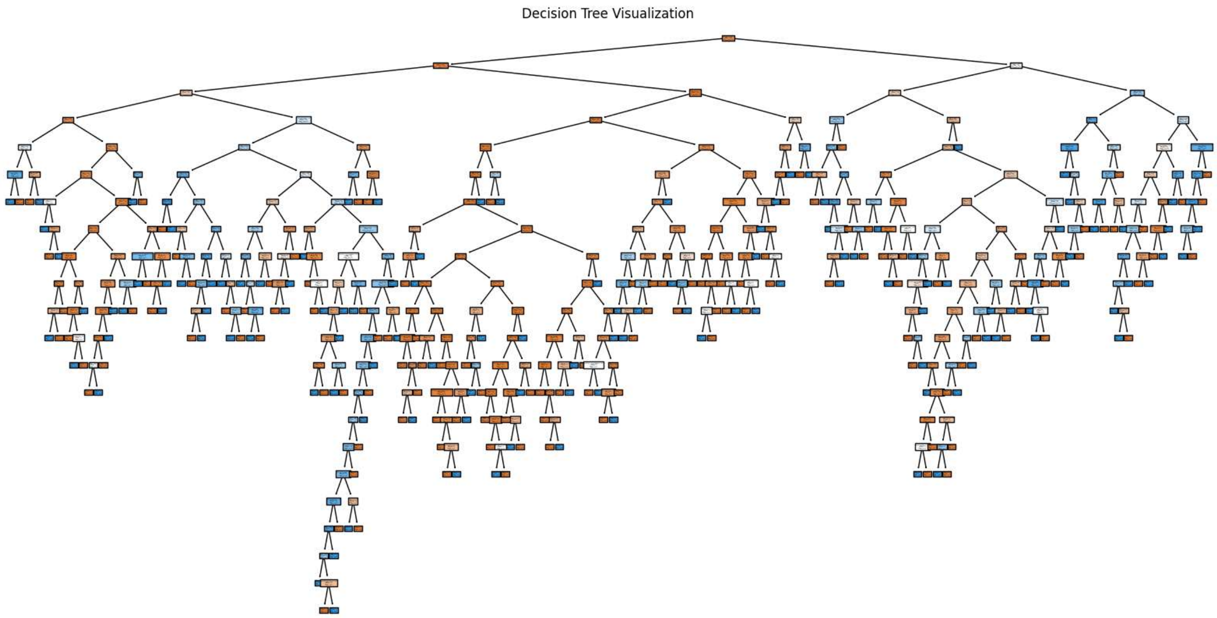
Out[8]:
```
DecisionTreeClassifier

DecisionTreeClassifier(random_state=42)
```

In [9]:
```
y_pred = dt.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

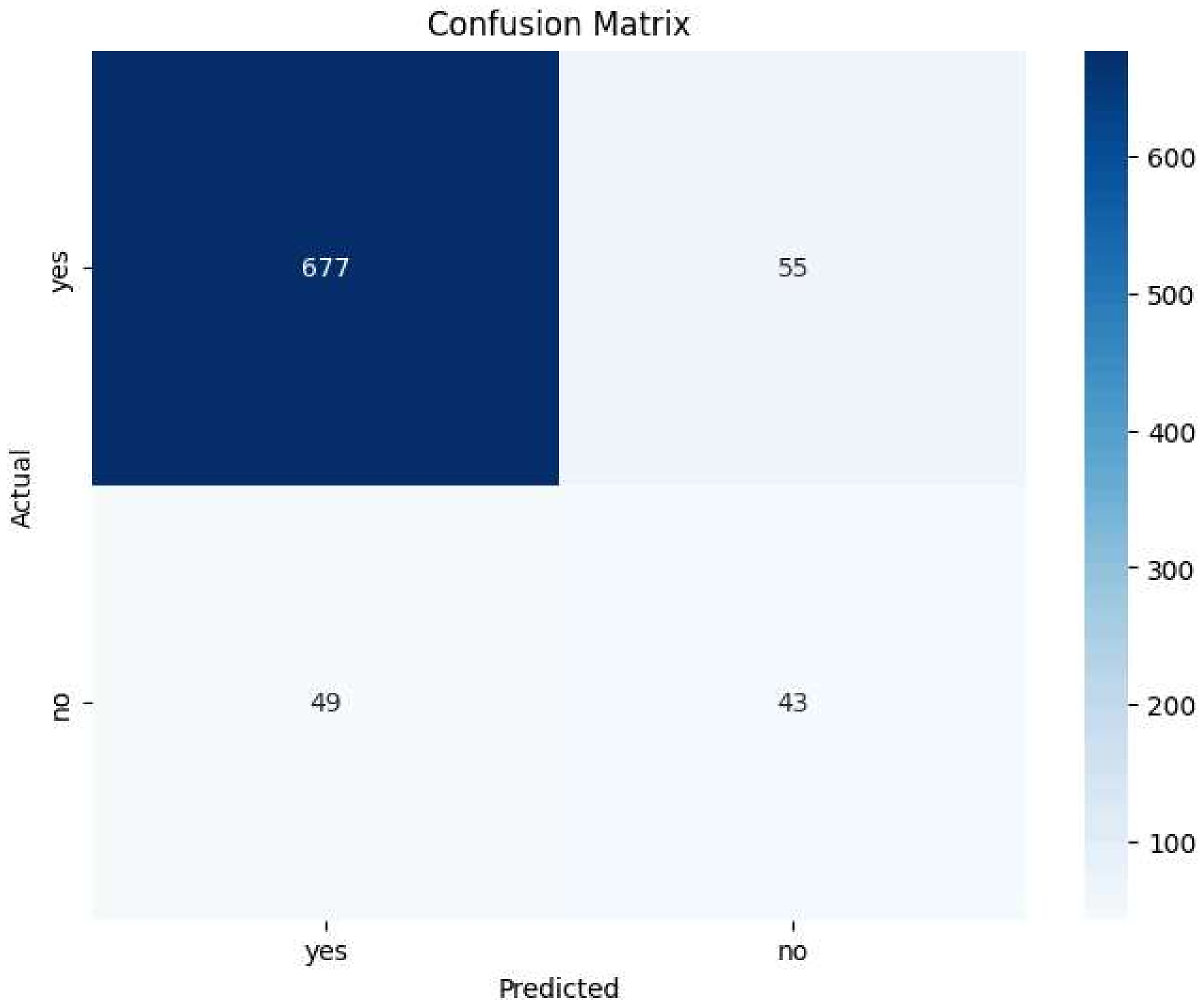
```
Accuracy: 0.87
              precision    recall  f1-score   support

          no       0.93      0.92      0.93       732
         yes       0.44      0.47      0.45        92

    accuracy                           0.87       824
   macro avg       0.69      0.70      0.69       824
weighted avg       0.88      0.87      0.88       824

[[677  55]
 [ 49  43]]
```

In [11]:
```
#our output suggests that our decision tree classifier achieved an accuracy of 0.87
#when predicting whether customers would subscribe to a term deposit based on their
#The precision and recall for both classes ('yes' and 'no') are also provided, indi

#This performance indicates a reasonably good fit for predicting customer behavior
#If you're satisfied with the accuracy and other metrics, it suggests that your dec
```

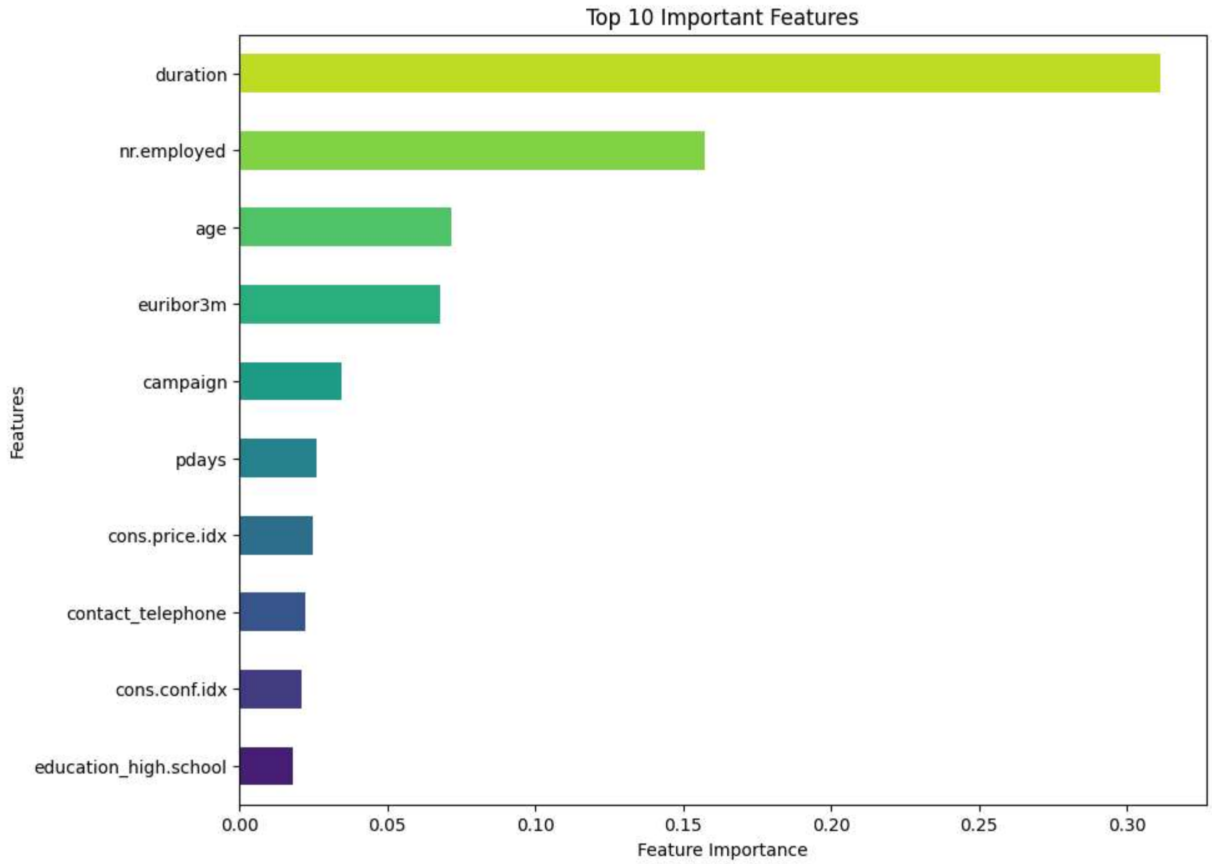
In [13]:
```
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt
cn = y.unique().astype(str)
plt.figure(figsize=(20, 10))
plot_tree(dt, filled=True, feature_names=X.columns, class_names=cn)
plt.title('Decision Tree Visualization')
plt.show()
```



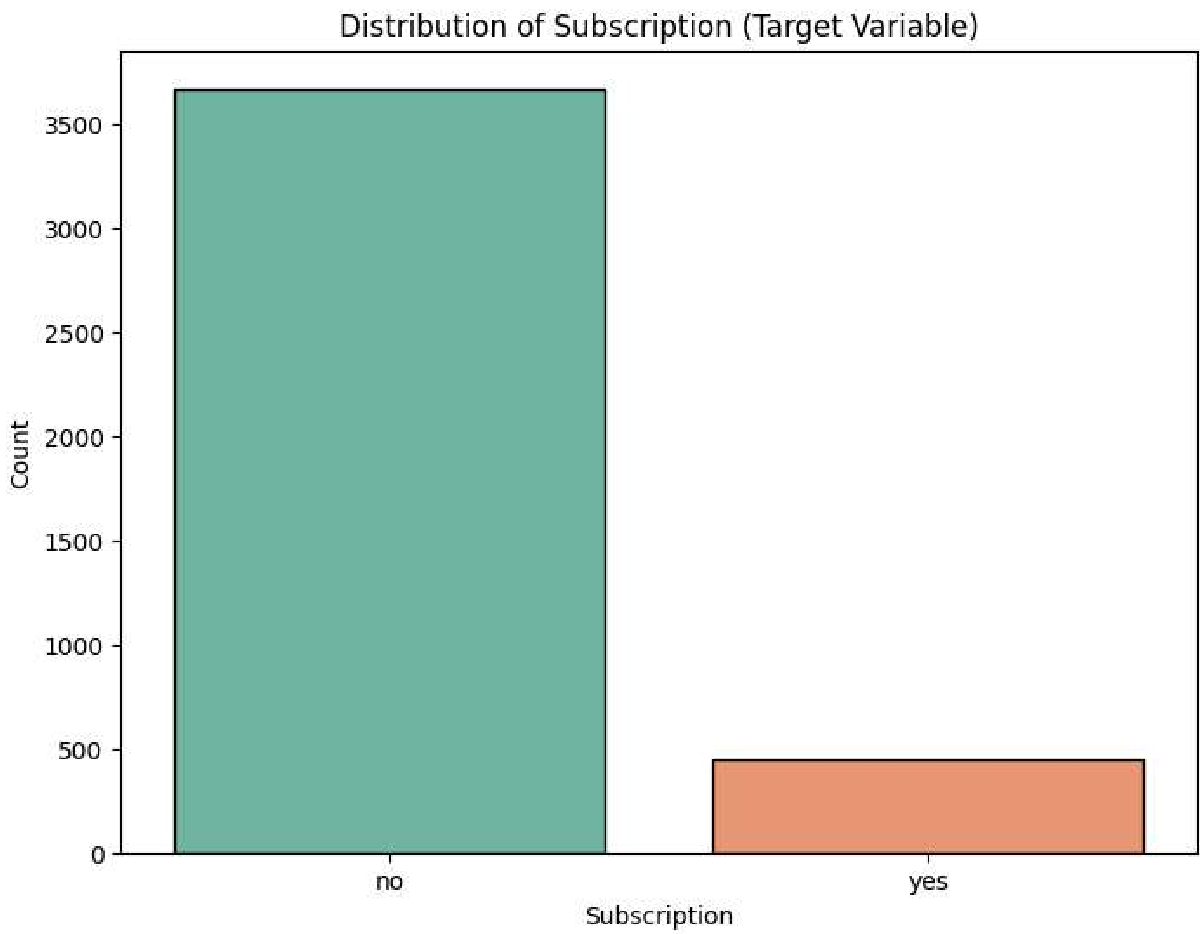
In [14]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns
cn = ['yes', 'no']
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, cmap='Blues', fmt='d', xticklabels=cn, yticklabels=cn)
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



In [15]:
```
plt.figure(figsize=(10, 8))
feat_importances = pd.Series(dt.feature_importances_, index=X.columns)
feat_importances.nlargest(10).sort_values().plot(kind='barh', color=sns.color_palet
plt.title('Top 10 Important Features')
plt.xlabel('Feature Importance')
plt.ylabel('Features')
plt.show()
```

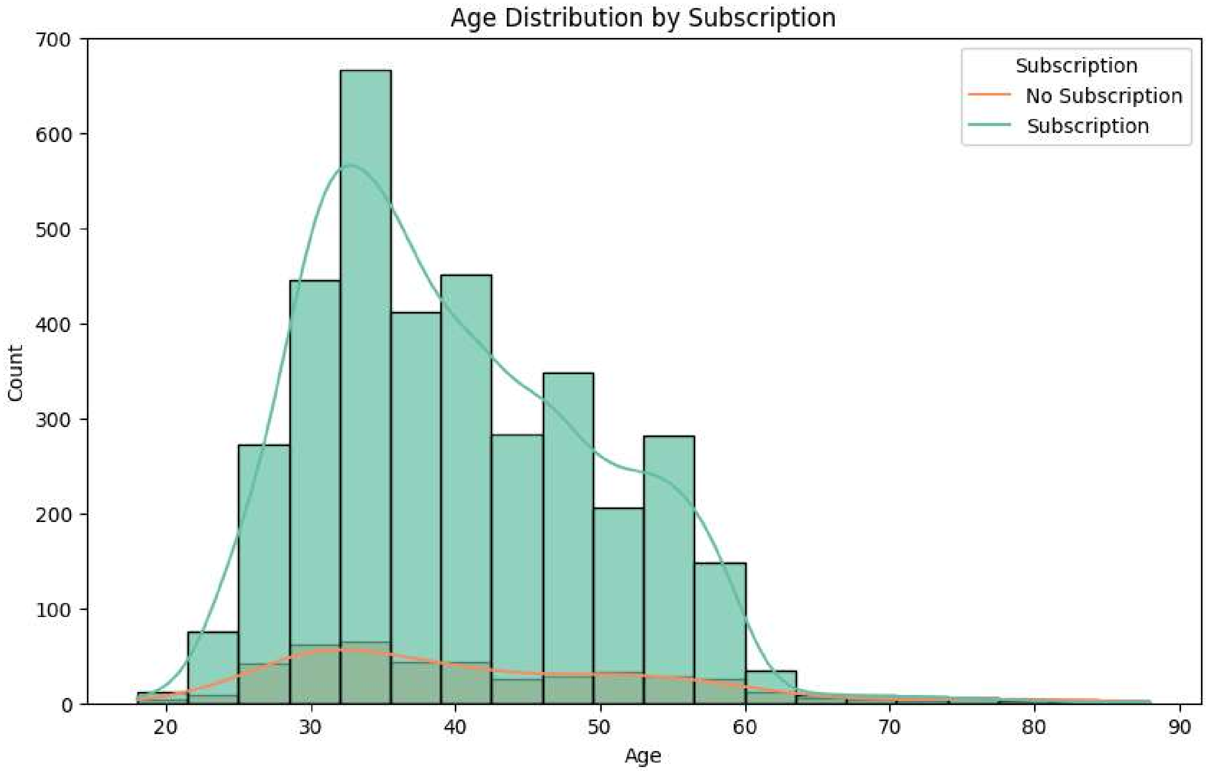


In [17]:
```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(8, 6))
sns.countplot(x='y', data=df, hue='y', palette='Set2', edgecolor='black', dodge=Fal
plt.title('Distribution of Subscription (Target Variable)')
plt.xlabel('Subscription')
plt.ylabel('Count')
plt.show()
```

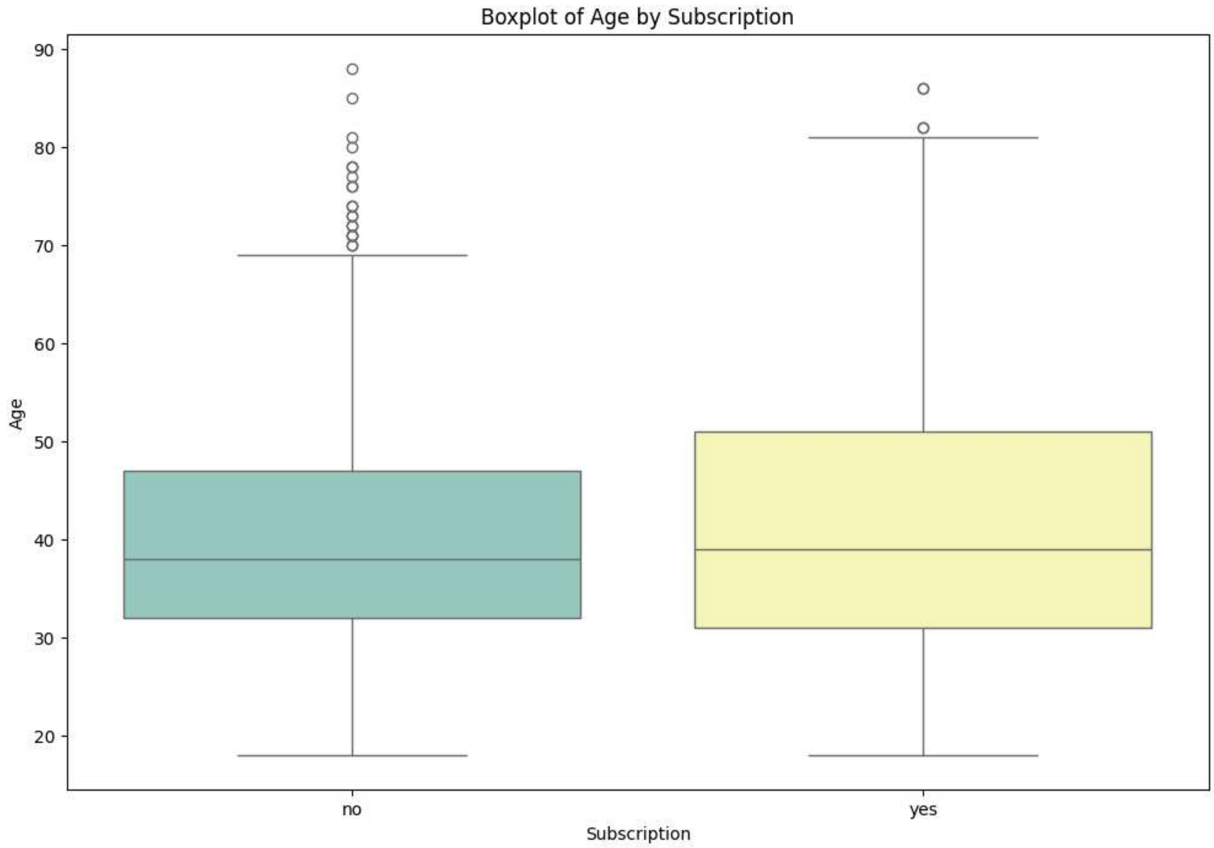


In [18]:
```
num_features = df.select_dtypes(include=['int64', 'float64']).columns
sns.pairplot(df[num_features])
plt.suptitle('Pairplot of Numerical Features', y=1.02)
plt.show()
```

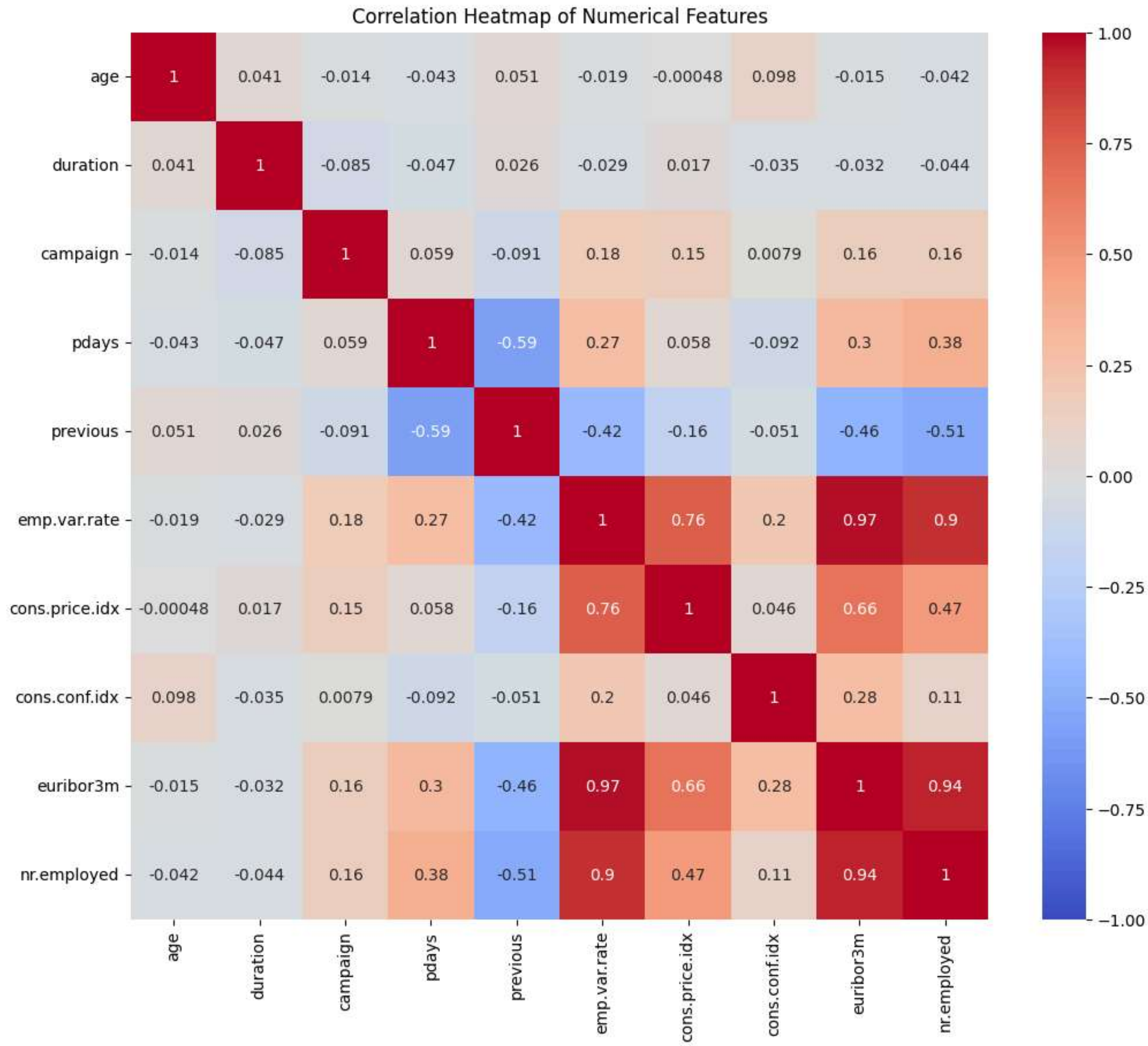
In [19]:
```
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='age', hue='y', bins=20, kde=True, palette='Set2', alpha=0.
plt.title('Age Distribution by Subscription')
plt.xlabel('Age')
plt.ylabel('Count')
plt.legend(labels=['No Subscription', 'Subscription'], title='Subscription')
plt.show()
```



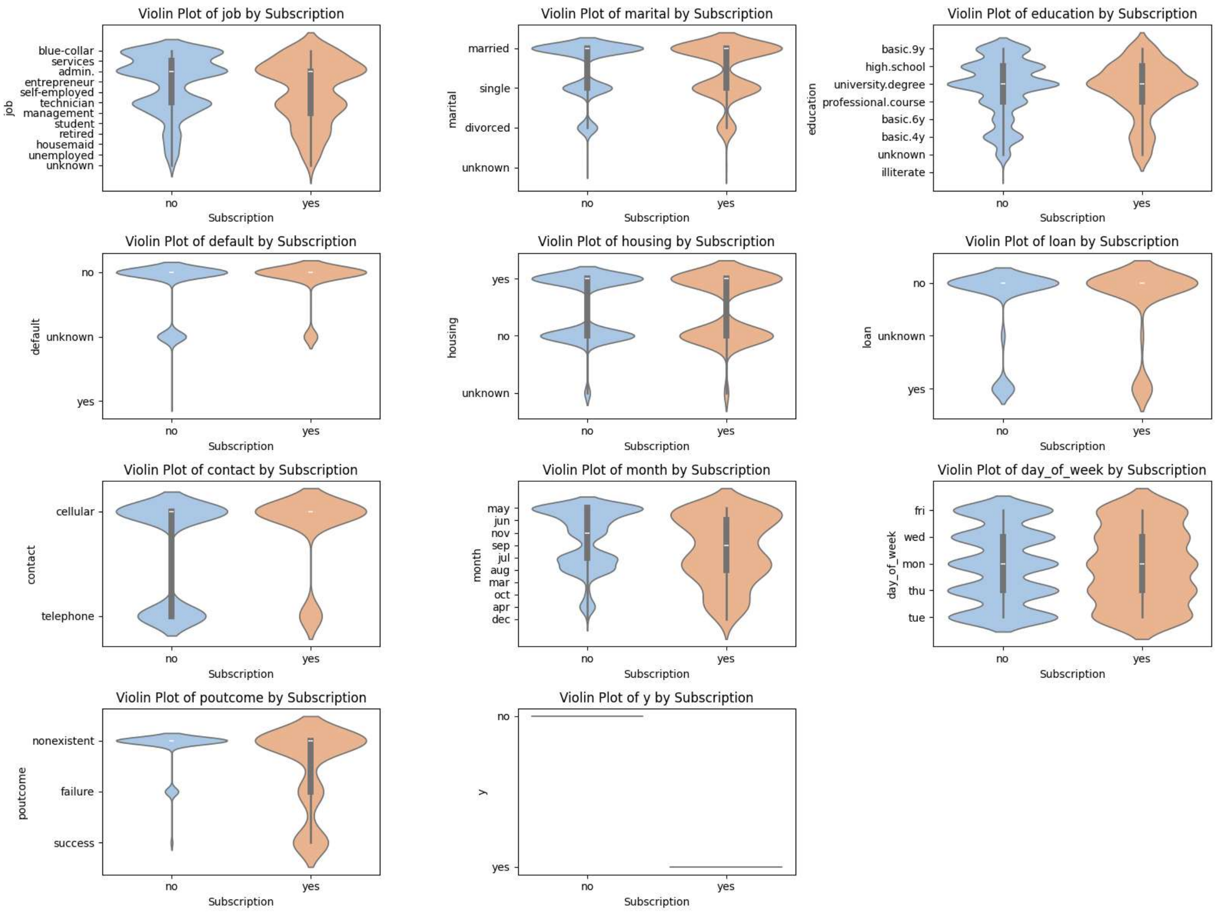
In [20]:
```
plt.figure(figsize=(12, 8))
sns.boxplot(data=df, x='y', y='age', palette='Set3', hue='y', dodge=False)
plt.title('Boxplot of Age by Subscription')
plt.xlabel('Subscription')
plt.ylabel('Age')
plt.show()
```



In [21]:
```
numeric_cols = df.select_dtypes(include=np.number)
plt.figure(figsize=(12, 10))
sns.heatmap(numeric_cols.corr(), annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Heatmap of Numerical Features')
plt.show()
```

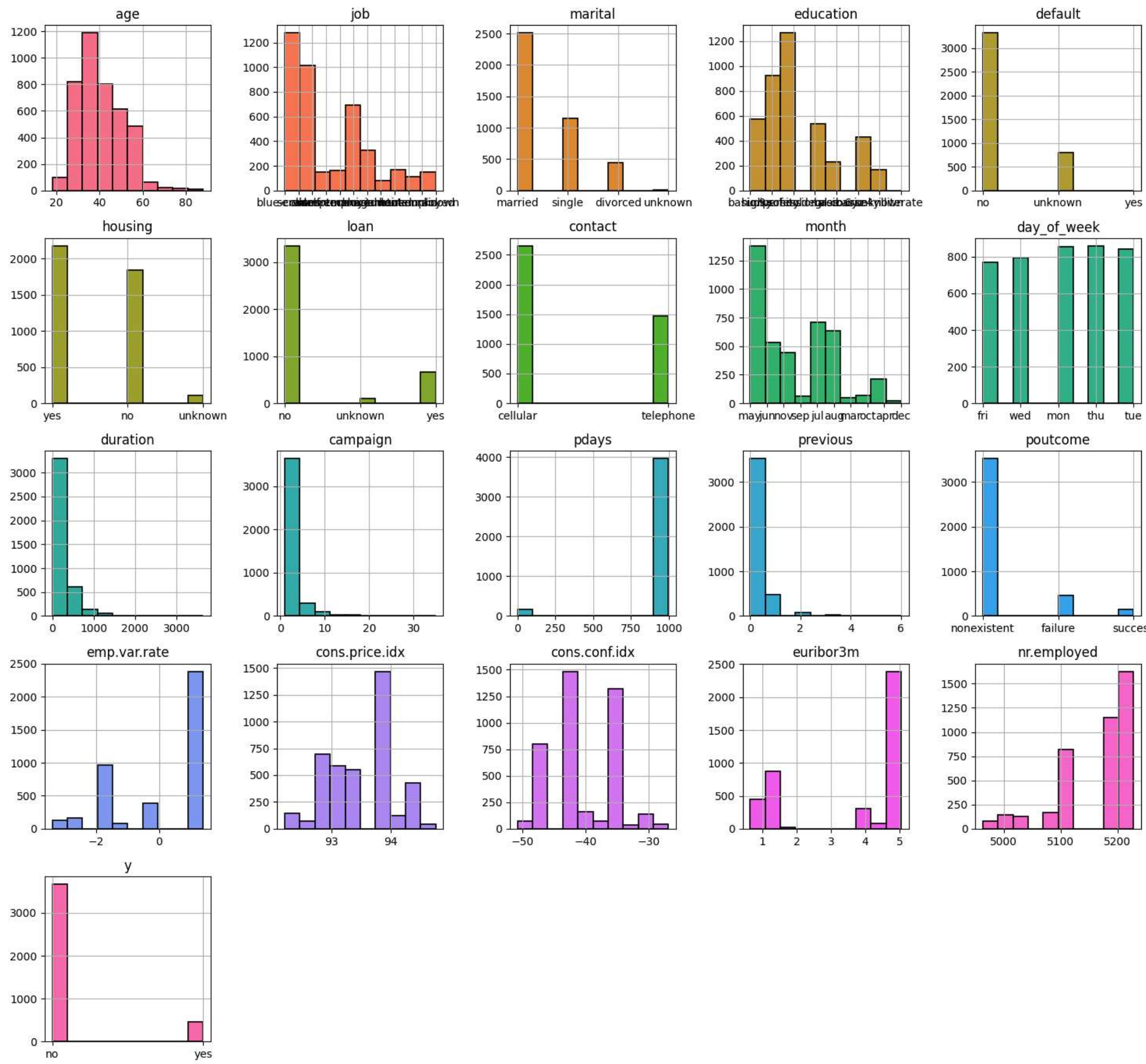


In [22]:
```
cat_features = df.select_dtypes(include=['object']).columns
num_cols = 3
num_rows = (len(cat_features) // num_cols) + 1  # Calculate number of rows needed

plt.figure(figsize=(16, 12))
for i, col in enumerate(cat_features, 1):
    plt.subplot(num_rows, num_cols, i)
    sns.violinplot(x='y', y=col, data=df, hue='y', palette='pastel', legend=False)
    plt.title(f'Violin Plot of {col} by Subscription')
    plt.xlabel('Subscription')
    plt.ylabel(col)

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



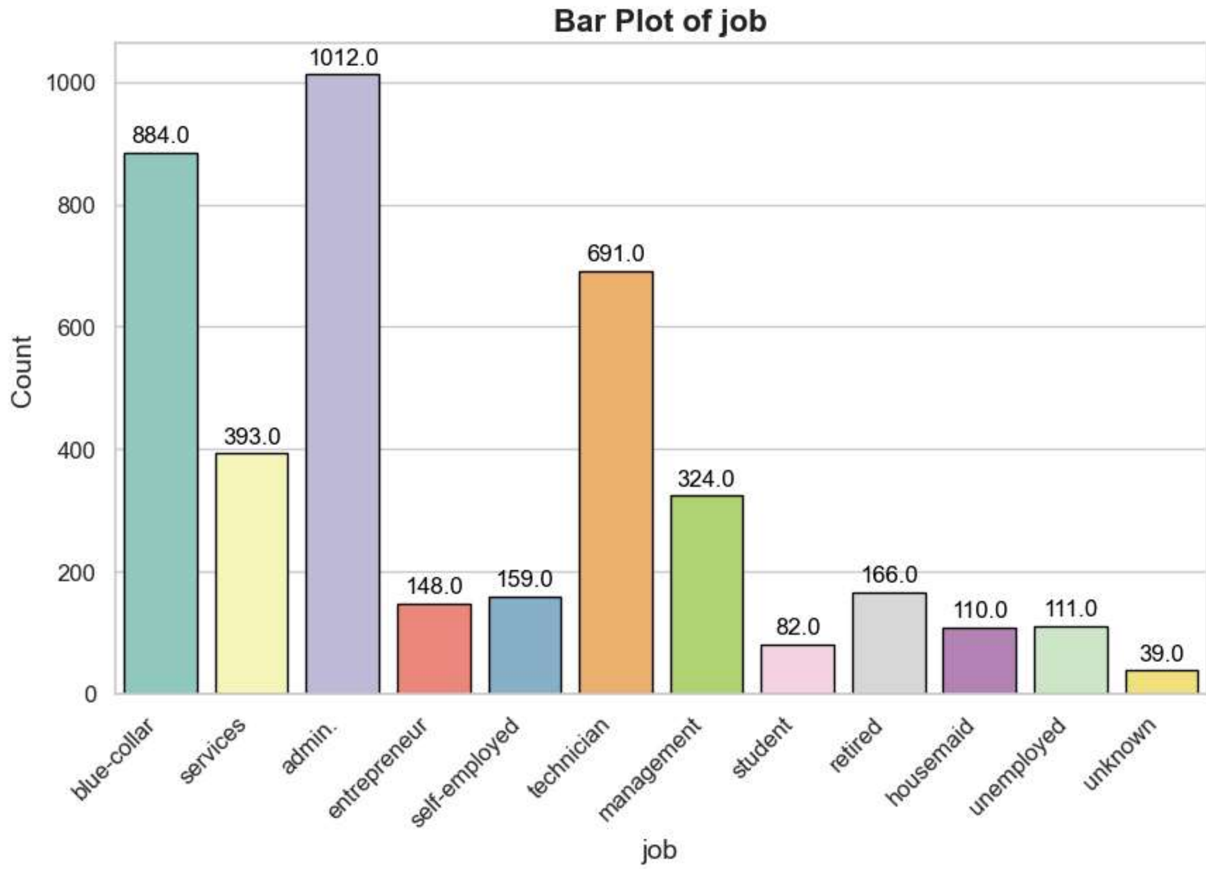
In [23]:
```
plt.figure(figsize=(15, 15))
color_palette = sns.color_palette("husl", len(df.columns))
for i, column in enumerate(df.columns):
    plt.subplot(5, 5, i + 1)
    df[column].hist(color=color_palette[i], edgecolor='black', linewidth=1.2)
    plt.title(column)
plt.suptitle('Distribution of Features', size=20)
plt.tight_layout(rect=[0, 0, 1, 0.95])  # Adjust layout to make space for the main
plt.show()
```

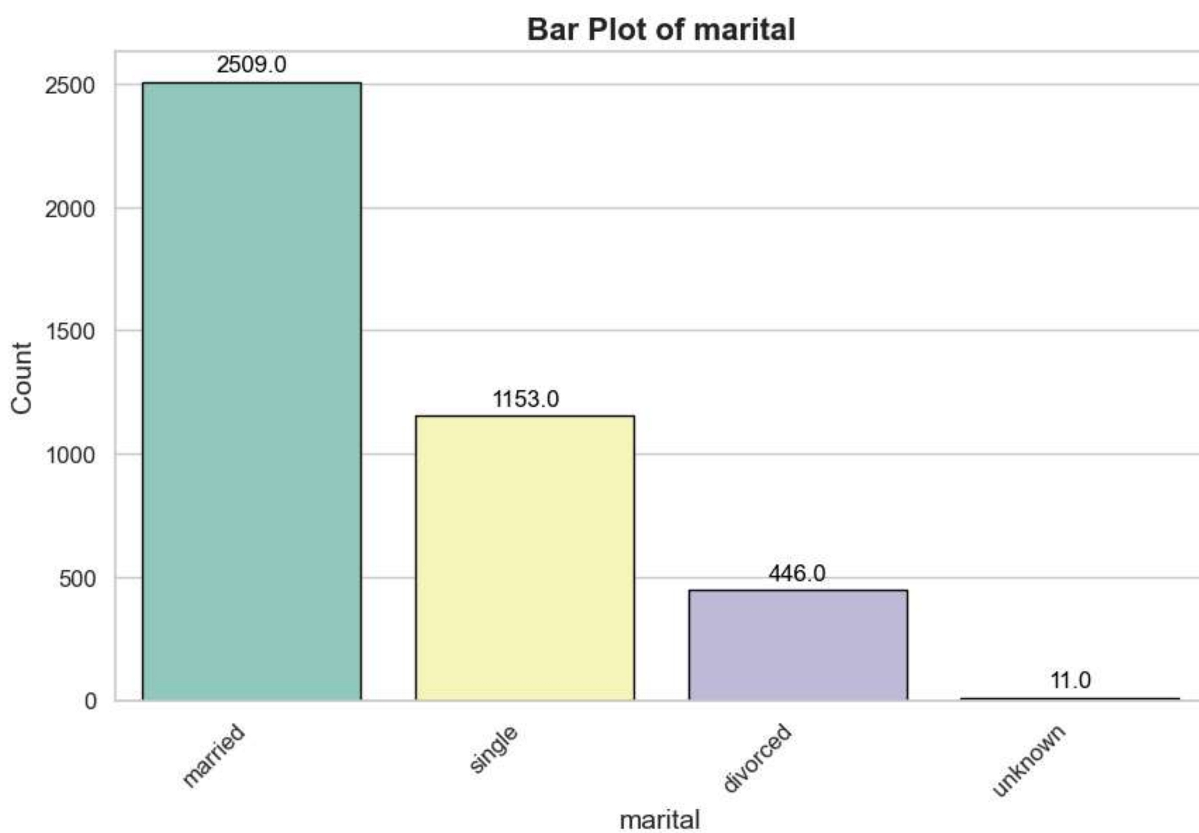
Distribution of Features

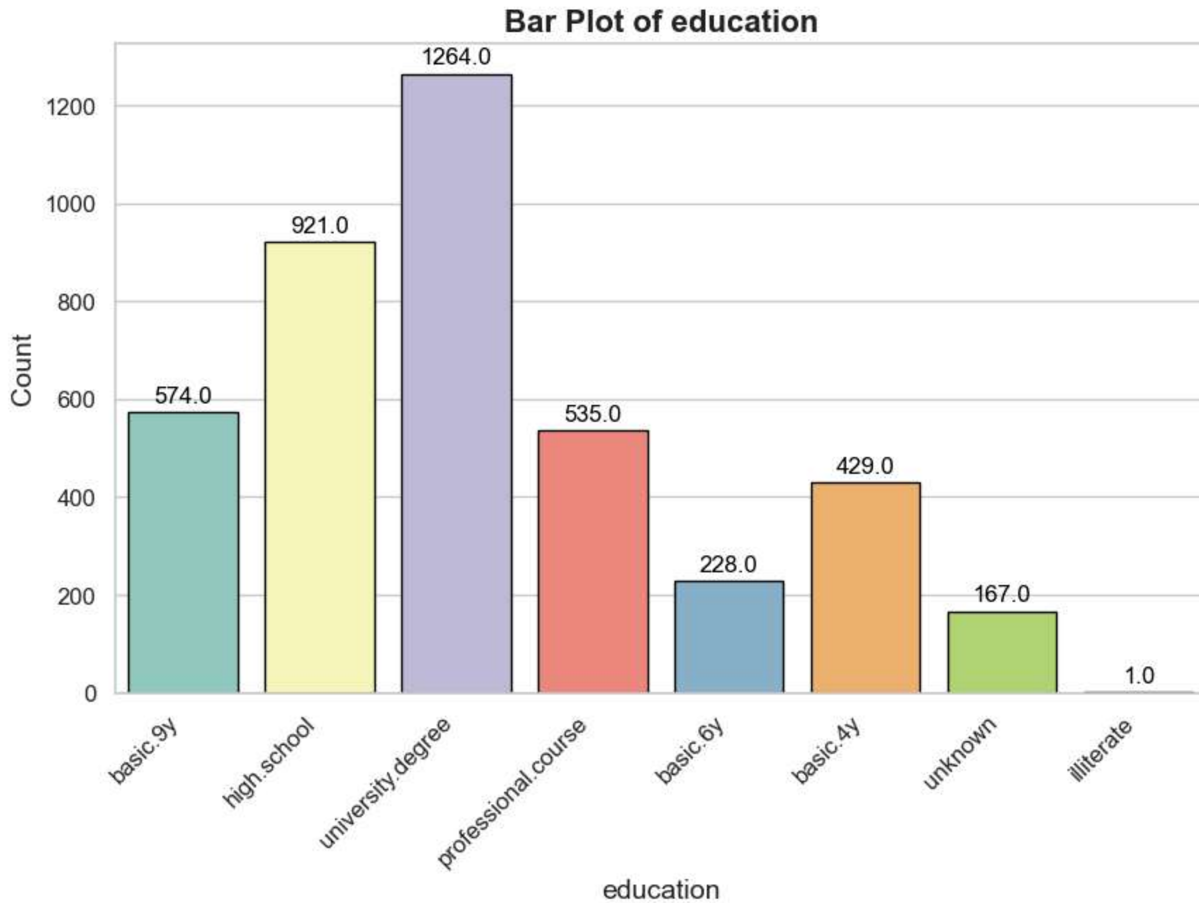


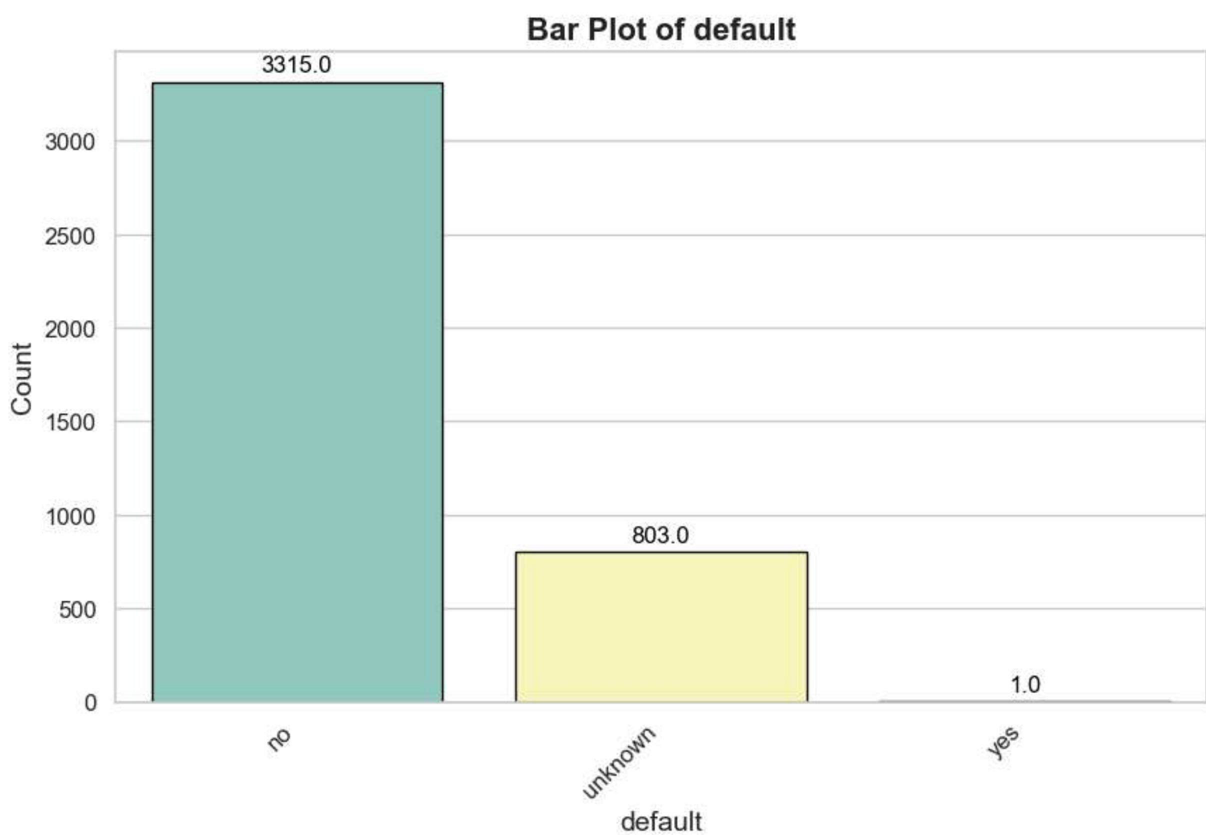
In [25]:
```
import seaborn as sns
import matplotlib.pyplot as plt
cat_cols = df.select_dtypes(include=['object']).columns
sns.set(style="whitegrid")
for feature in cat_cols:
    plt.figure(figsize=(10, 6))  # Adjust the figure size as needed
    barplot = sns.countplot(x=feature, hue=feature, data=df, palette='Set3', edgeco


    plt.title(f'Bar Plot of {feature}', fontsize=16, fontweight='bold')
    plt.xlabel(feature, fontsize=14)
    plt.ylabel('Count', fontsize=14)
    plt.xticks(rotation=45, fontsize=12, ha='right')
    plt.yticks(fontsize=12)


    for p in barplot.patches:
        barplot.annotate(f'{p.get_height()}', (p.get_x() + p.get_width() / 2., p.ge
                         ha='center', va='baseline', fontsize=12, color='black', xy
```

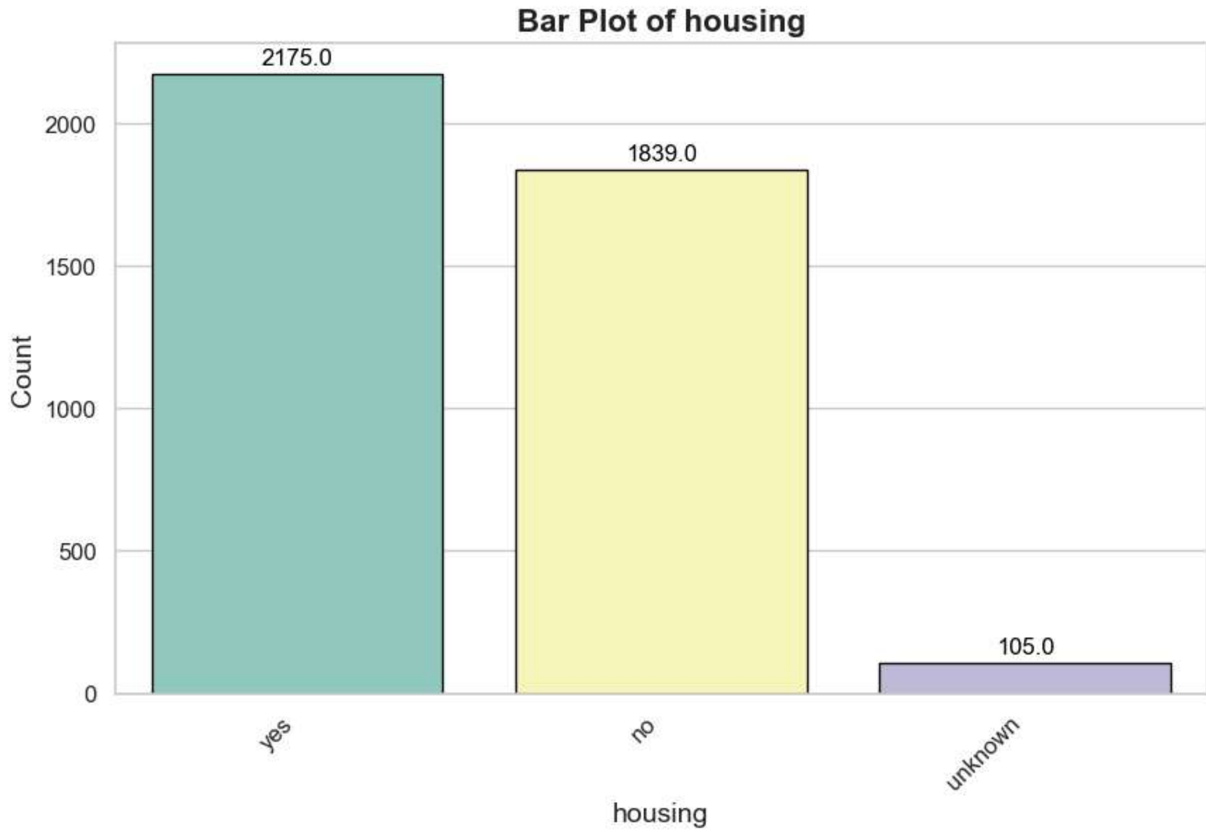
```
                                textcoords='offset points')

    plt.show()
```

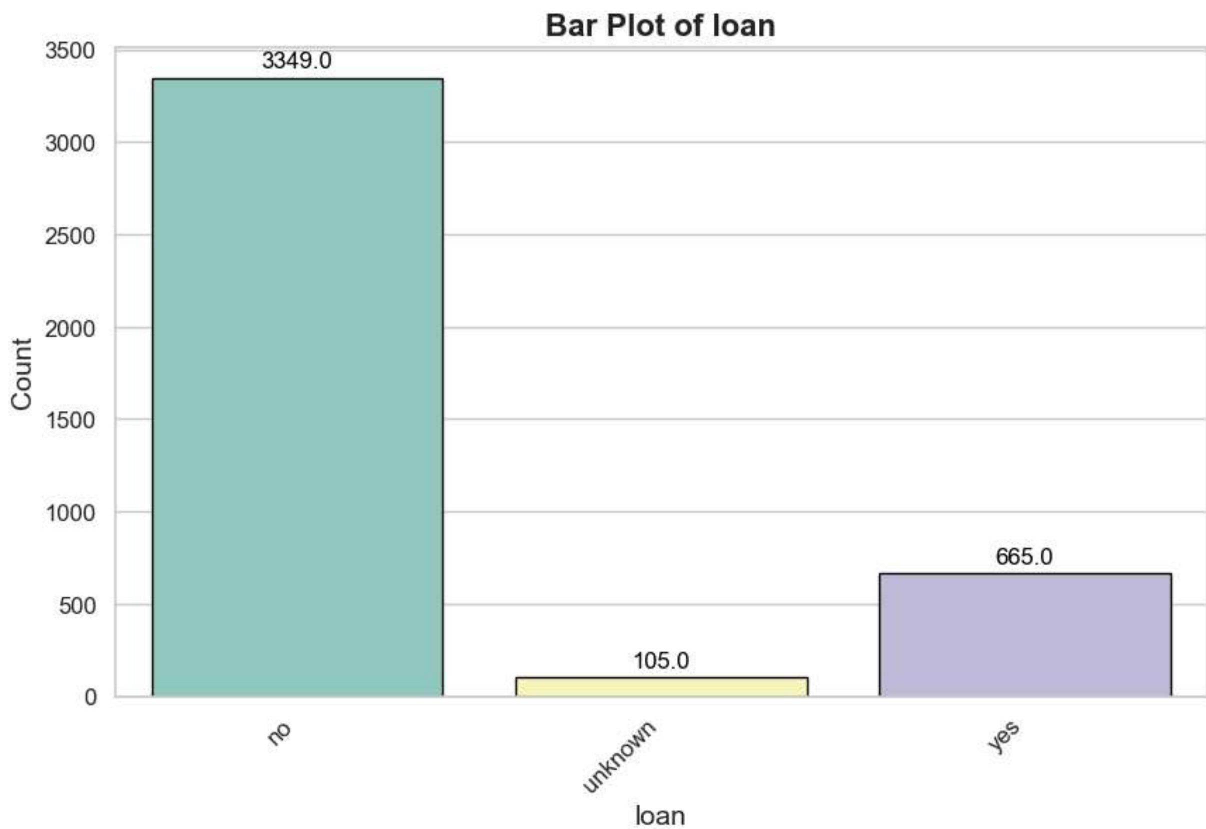


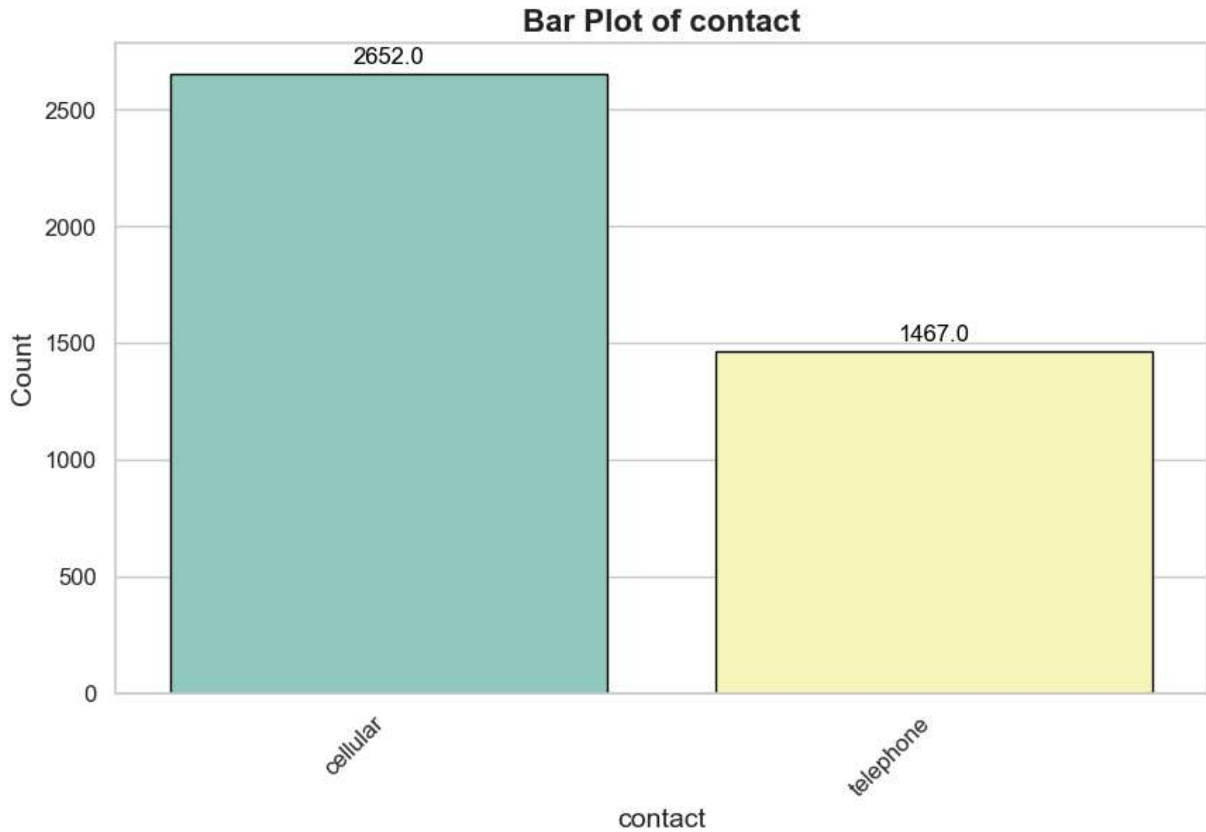


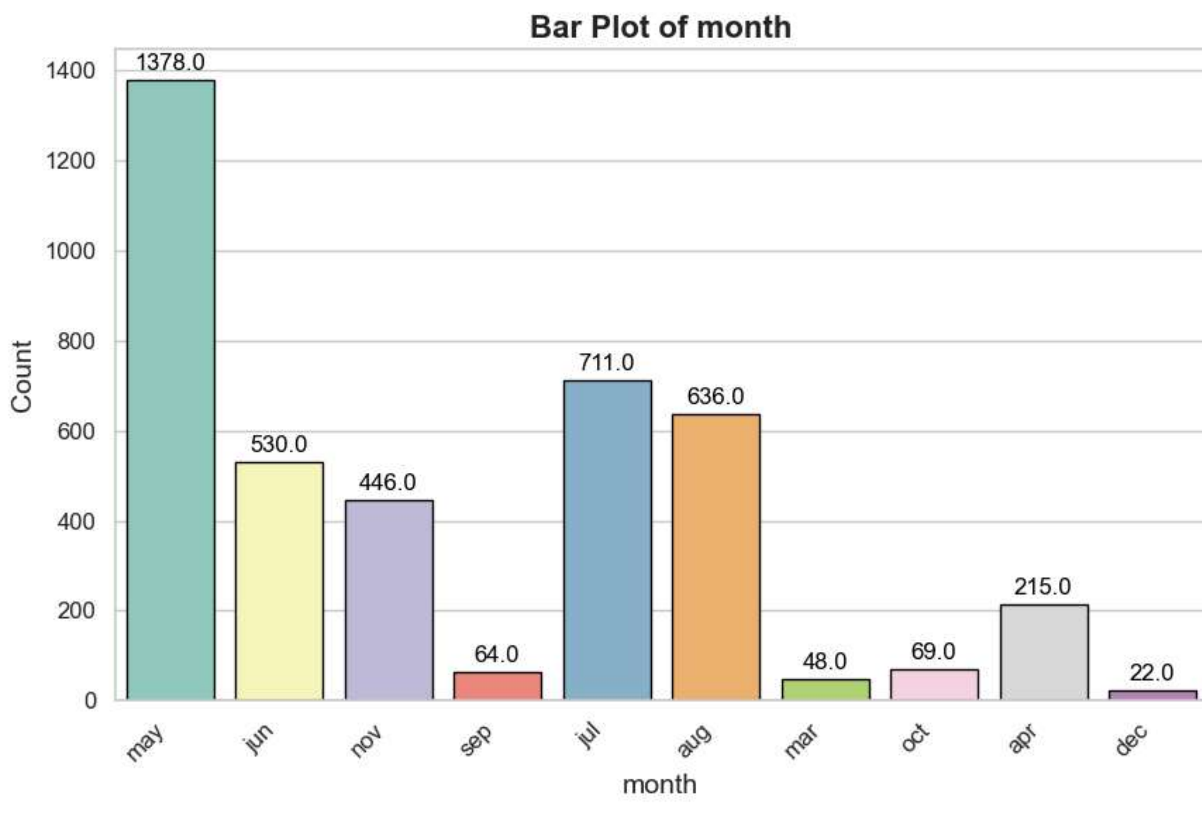

Bar Plot of education
Count
education
1264.0
921.0
574.0
535.0
429.0
228.0
167.0
1.0
0
200
400
600
800
1000
1200
basic.9y
high.school
university.degree
professional.course
basic.6y
basic.4y
unknown
illiterate


Bar Plot of default
Count
default
3315.0
803.0
1.0
0
500
1000
1500
2000
2500
3000
no
unknown
yes

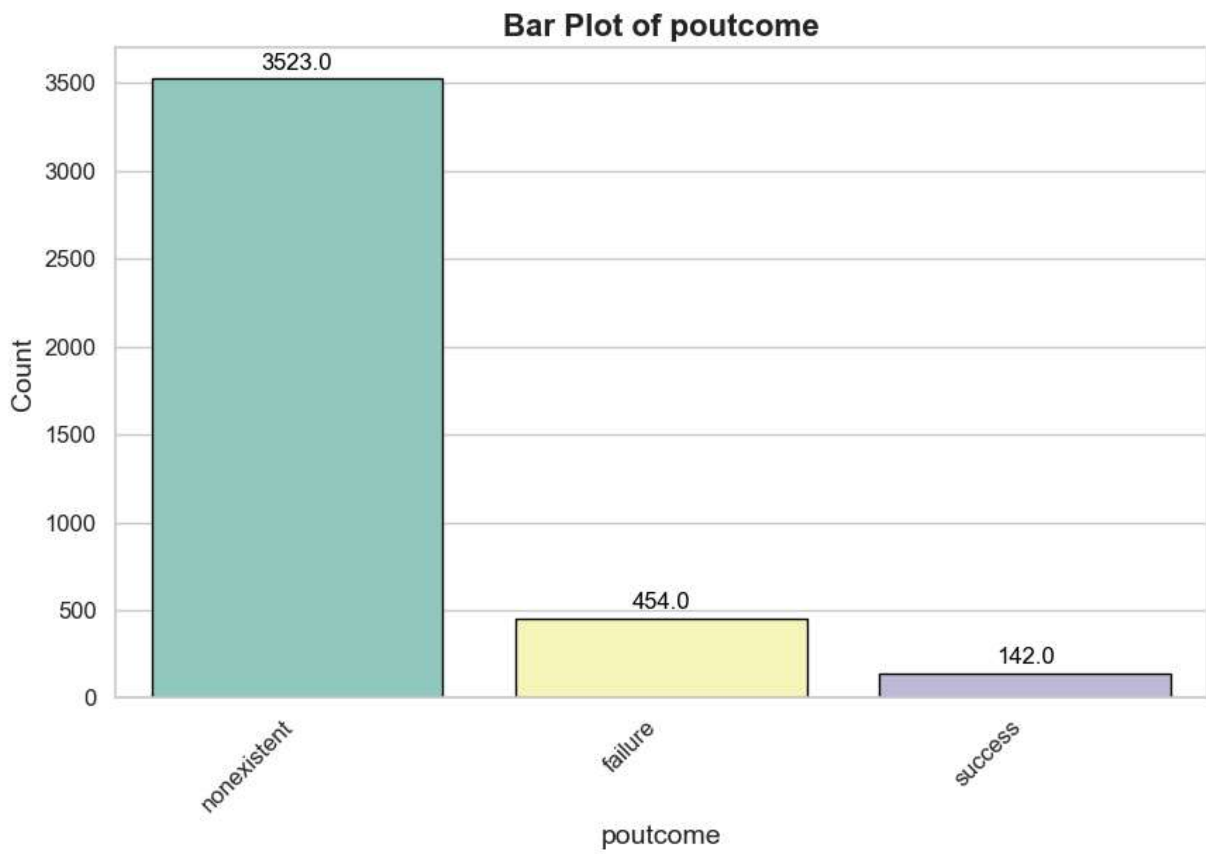

Bar Plot of housing
2175.0
1839.0
105.0
2000
1500
1000
500
0
Count
yes
no
unknown
housing

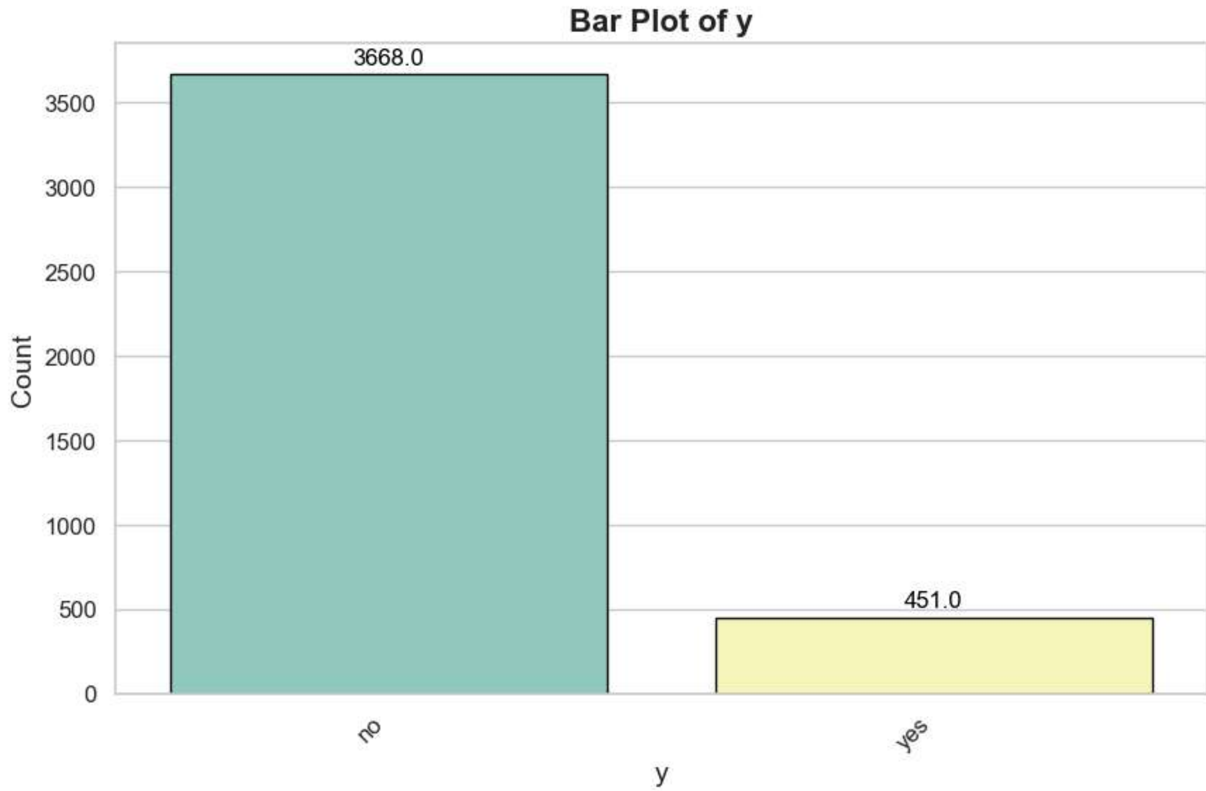

Bar Plot of loan
3349.0
105.0
665.0
3500
3000
2500
2000
1500
1000
500
0
Count
no
unknown
yes
loan


Bar Plot of contact
2652.0
1467.0
Count
0
500
1000
1500
2000
2500
cellular
telephone
contact

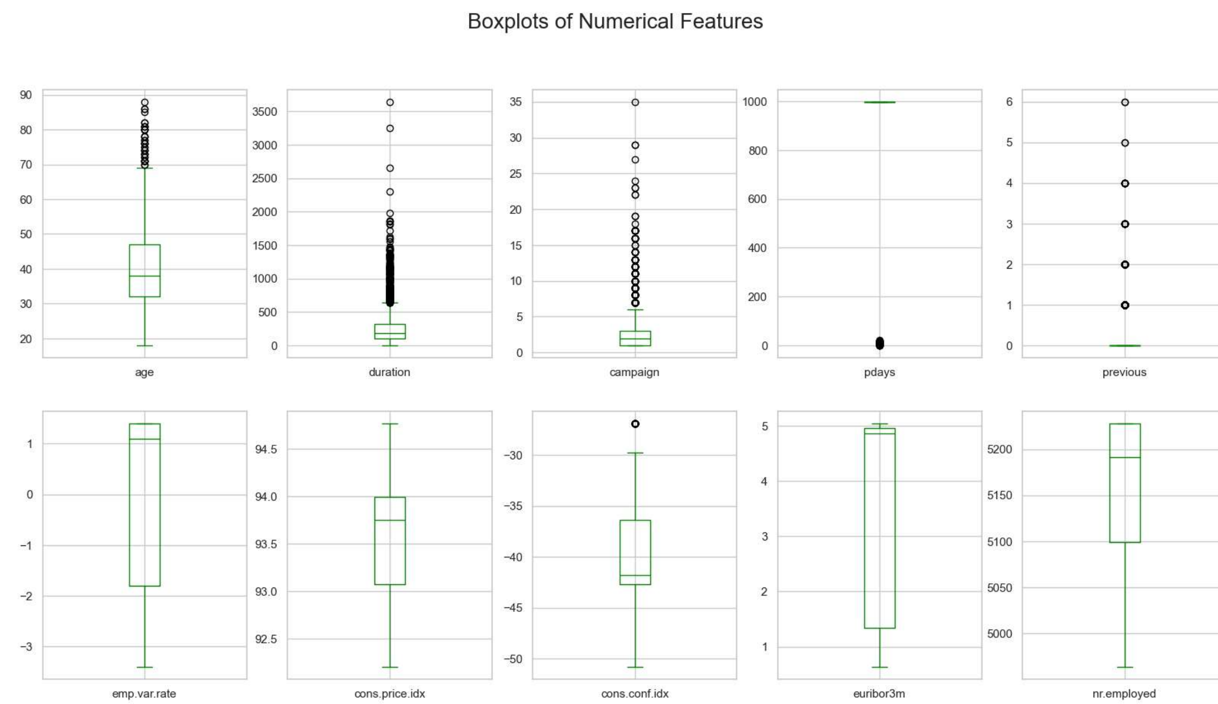

Bar Plot of month
1378.0
530.0
446.0
64.0
711.0
636.0
48.0
69.0
215.0
22.0
Count
0
200
400
600
800
1000
1200
1400
may
jun
nov
sep
jul
aug
mar
oct
apr
dec
month


Bar Plot of day_of_week
768.0
795.0
855.0
860.0
841.0
800
600
400
200
0
Count
fri
wed
mon
thu
tue
day_of_week


Bar Plot of poutcome
3523.0
454.0
142.0
3500
3000
2500
2000
1500
1000
500
0
Count
nonexistent
failure
success
poutcome



In [29]:
```
df.plot(kind='box', subplots=True, layout=(2,5),figsize=(20,10),color='green')
plt.suptitle('Boxplots of Numerical Features', fontsize=20)
plt.show()
```



In [ ]: