In [ ]:

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

In [2]:
```
df = pd.read_csv('C:\\Users\\lenovo\\Desktop\\courses\\Task3\\bank-additional.csv',
```

In [3]:
```
df
```

Out[3]:

	age	job	marital	education	default	housing	loan	contact	month
0	30	blue-collar	married	basic.9y	no	yes	no	cellular	may
1	39	services	single	high.school	no	no	no	telephone	may
2	25	services	married	high.school	no	yes	no	telephone	jun
3	38	services	married	basic.9y	no	unknown	unknown	telephone	jun
4	47	admin.	married	university.degree	no	yes	no	cellular	nov
...	...	...	...	...	...	...	...	...	...
4114	30	admin.	married	basic.6y	no	yes	yes	cellular	jul
4115	39	admin.	married	high.school	no	yes	no	telephone	jul
4116	27	student	single	high.school	no	no	no	cellular	may
4117	58	admin.	married	high.school	no	no	no	cellular	aug
4118	34	management	single	high.school	no	yes	no	cellular	nov

4119 rows × 21 columns

In [4]:
```
df.head()
```

Out[4]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_
0	30	blue-collar	married	basic.9y	no	yes	no	cellular	may	
1	39	services	single	high.school	no	no	no	telephone	may	
2	25	services	married	high.school	no	yes	no	telephone	jun	
3	38	services	married	basic.9y	no	unknown	unknown	telephone	jun	
4	47	admin.	married	university.degree	no	yes	no	cellular	nov	

5 rows × 21 columns

In [5]:
```
df.columns
```

Out[5]:
```
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', 'target'],
      dtype='object')
```

In [6]:
```
df.dtypes
```

Out[6]:
```
age                int64
job               object
marital           object
education         object
default           object
housing           object
loan              object
contact           object
month             object
day_of_week       object
duration           int64
campaign           int64
pdays              int64
previous           int64
poutcome          object
emp.var.rate     float64
cons.price.idx   float64
cons.conf.idx    float64
euribor3m        float64
nr.employed      float64
target            object
dtype: object
```

In [7]:
```
missing_values = df.isnull().sum()
print("Valeurs manquantes dans les colonnes :")
print(missing_values)
```

```
Valeurs manquantes dans les colonnes :
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
target            0
dtype: int64
```

In [8]:
```
df.duplicated().sum()
```

Out[8]:
```
0
```

In [9]:
```
 df.isna().sum()
```

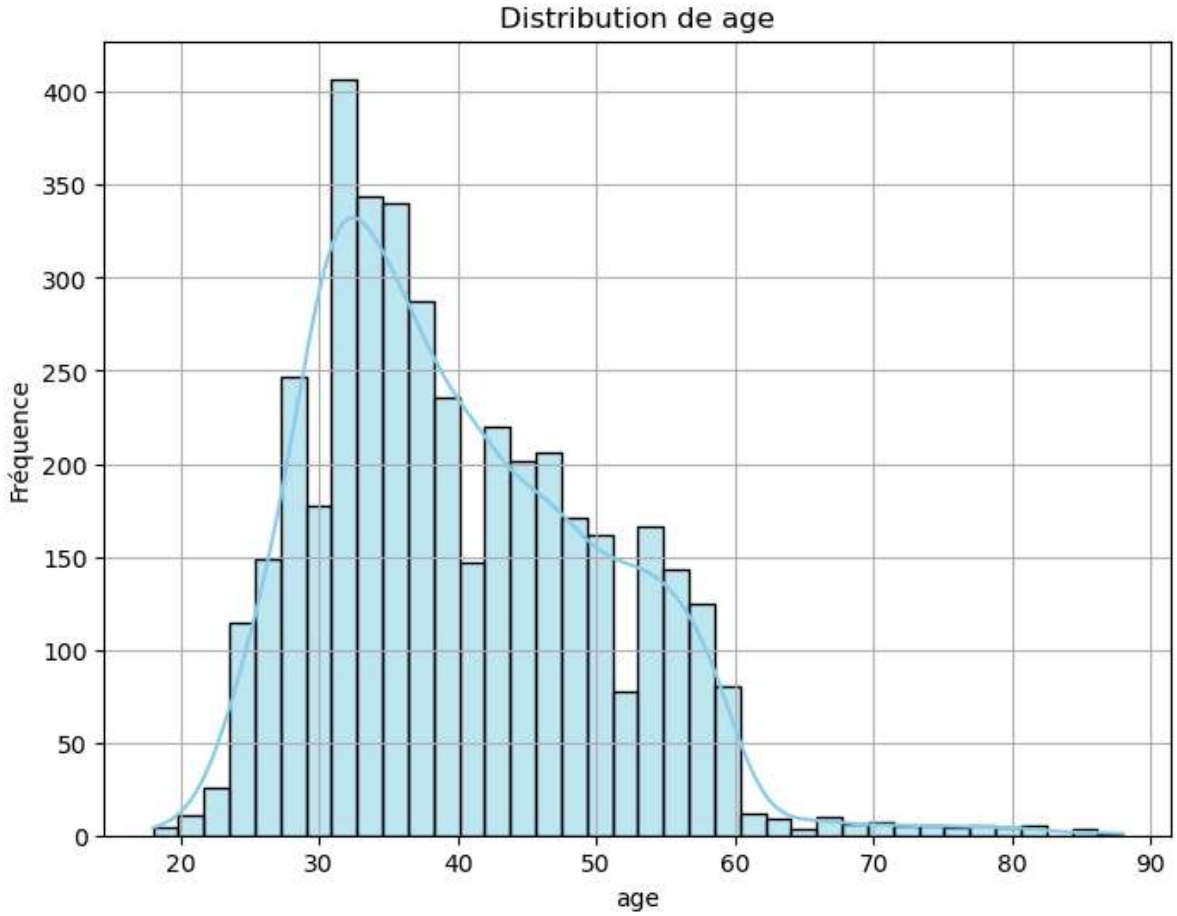
Out[9]:
```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
target            0
dtype: int64
```

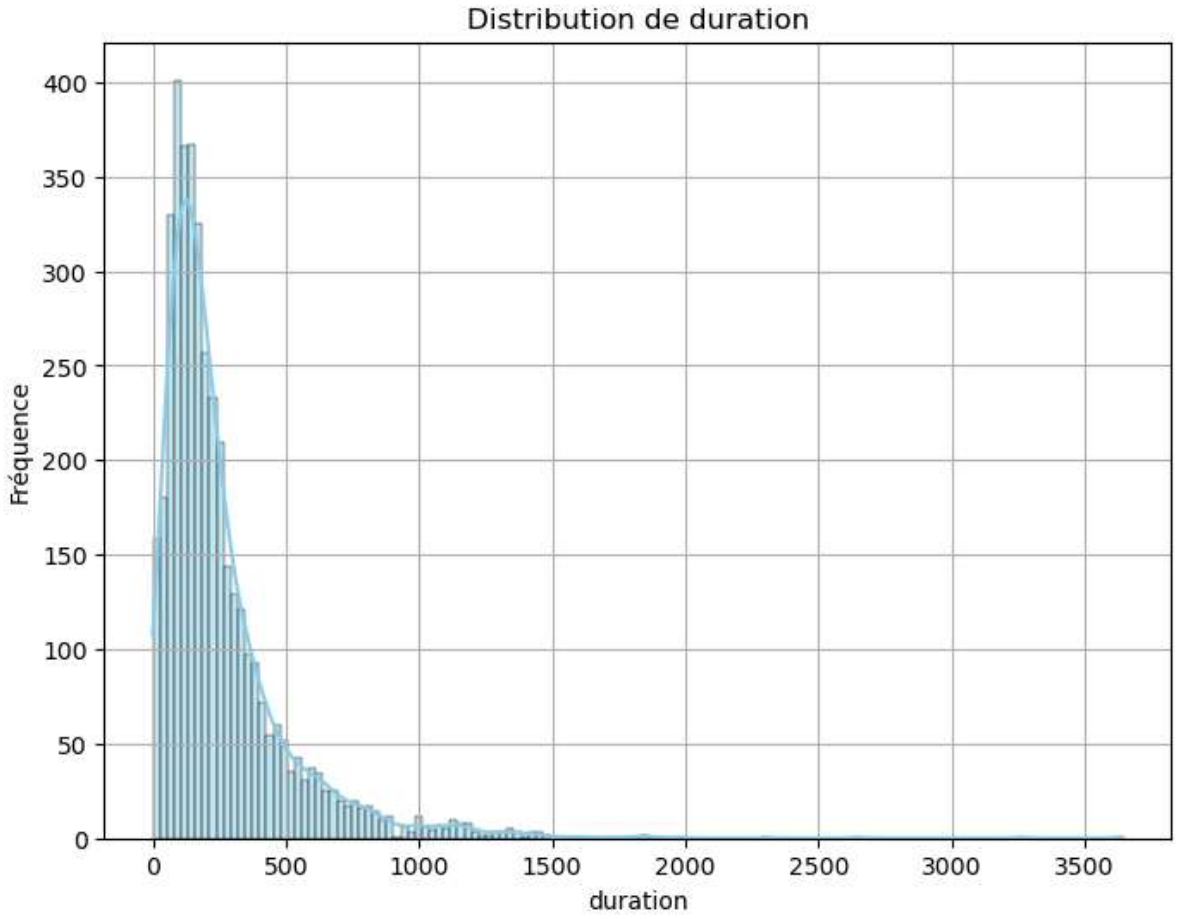
In [10]:
```
categorical_columns = df.select_dtypes(include=['object']).columns

numeric_columns = df.select_dtypes(include=['int64', 'float64']).columns

df_categorical = df[categorical_columns]
df_numeric = df[numeric_columns]

print("Dataframe des colonnes catégorielles :")
print(df_categorical.head())

print("\nDataframe des colonnes numériques :")
print(df_numeric.head())
```

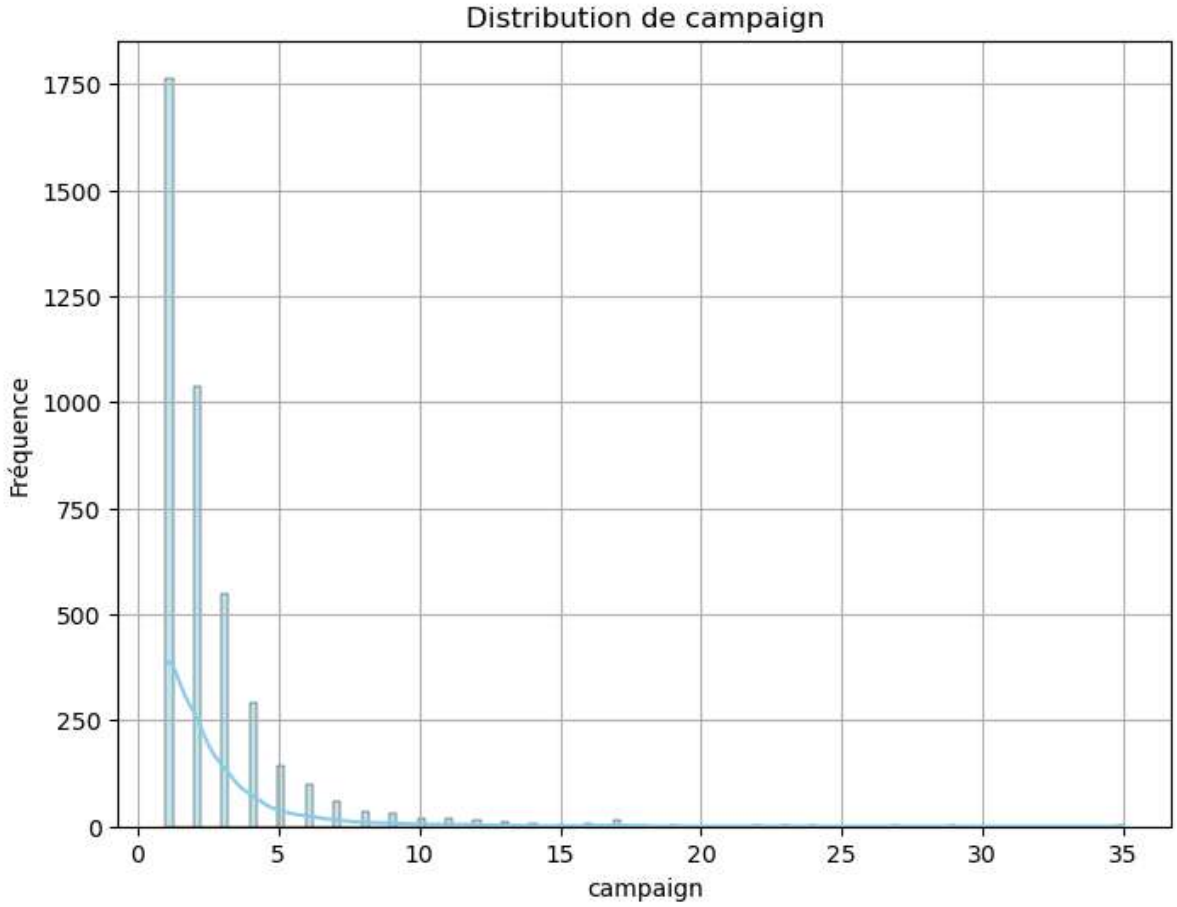
```
Dataframe des colonnes catégorielles :
           job  marital          education default  housing     loan  \
0  blue-collar  married           basic.9y      no      yes       no
1     services   single        high.school      no       no       no
2     services  married        high.school      no      yes       no
3     services  married           basic.9y      no  unknown  unknown
4       admin.  married  university.degree      no      yes       no

     contact month day_of_week     poutcome target
0   cellular   may         fri  nonexistent     no
1  telephone   may         fri  nonexistent     no
2  telephone   jun         wed  nonexistent     no
3  telephone   jun         fri  nonexistent     no
4   cellular   nov         mon  nonexistent     no

Dataframe des colonnes numériques :
   age  duration  campaign  pdays  previous  emp.var.rate  cons.price.idx  \
0   30       487         2    999         0          -1.8          92.893
1   39       346         4    999         0           1.1          93.994
2   25       227         1    999         0           1.4          94.465
3   38        17         3    999         0           1.4          94.465
4   47        58         1    999         0          -0.1          93.200

   cons.conf.idx  euribor3m  nr.employed
0          -46.2      1.313       5099.1
1          -36.4      4.855       5191.0
2          -41.8      4.962       5228.1
3          -41.8      4.959       5228.1
4          -42.0      4.191       5195.8
```

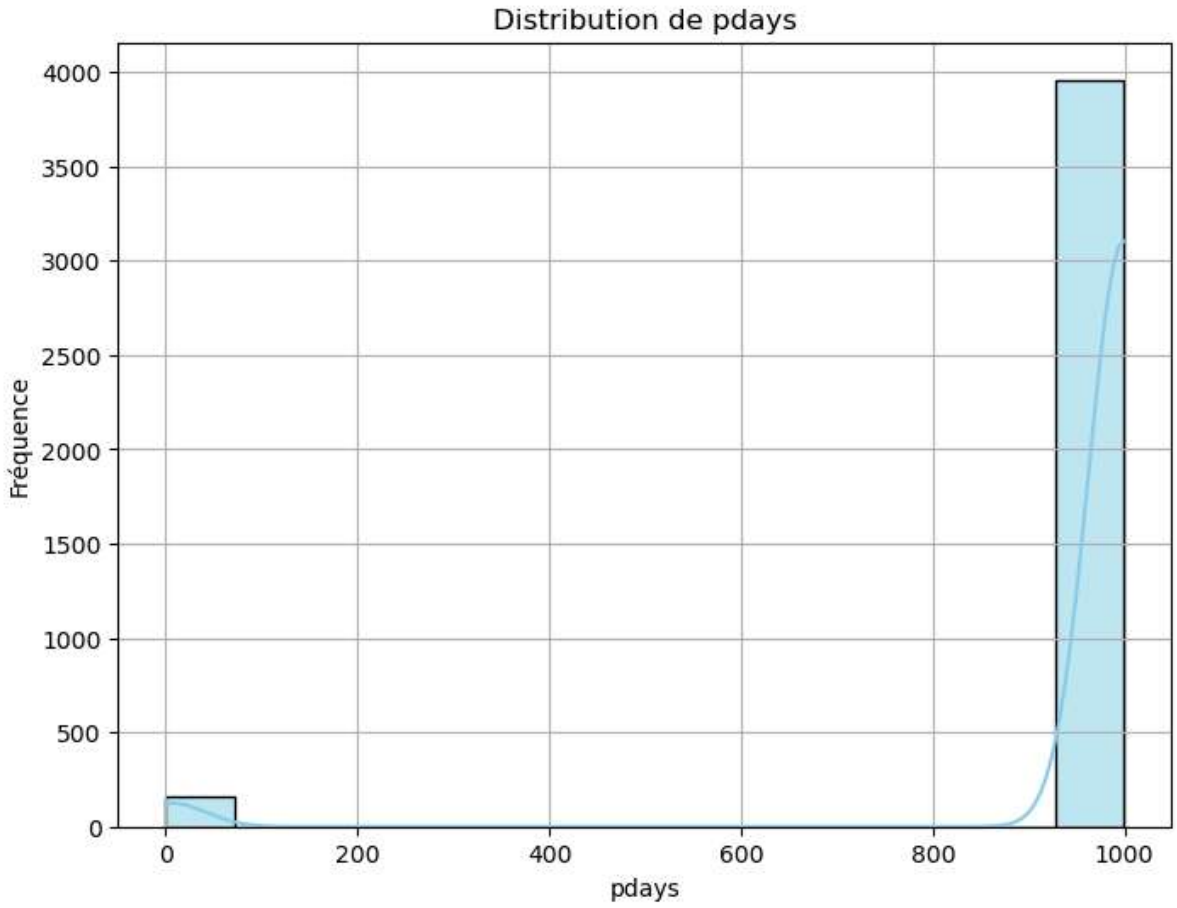
In [11]:

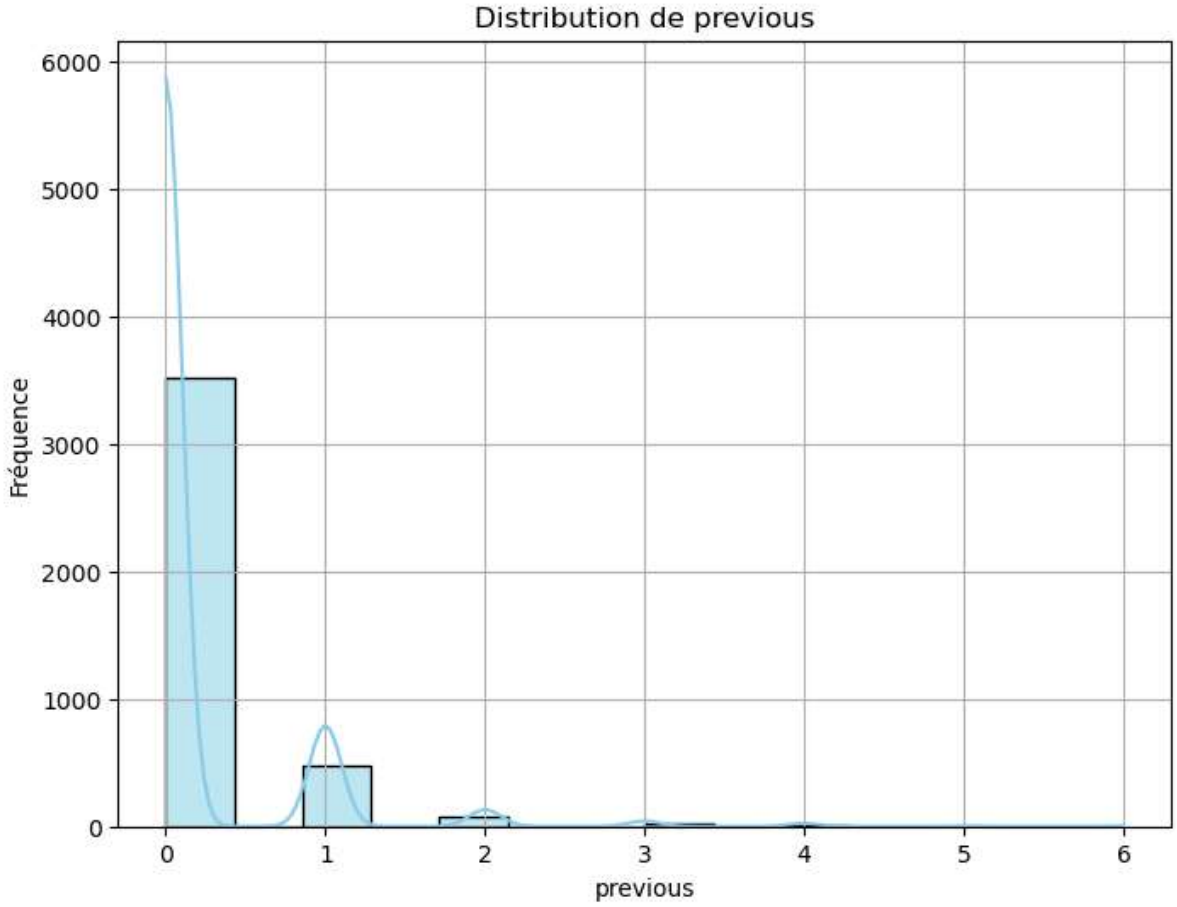
```python
significant_numeric_columns = ['age', 'duration', 'campaign', 'pdays', 'previous',
                               'emp.var.rate', 'cons.price.idx', 'cons.conf.idx',
                               'euribor3m', 'nr.employed']

for column in significant_numeric_columns:
    plt.figure(figsize=(8, 6))
    sns.histplot(df_numeric[column], kde=True, color='skyblue', edgecolor='black')
    plt.title(f'Distribution de {column}')
    plt.xlabel(column)
    plt.ylabel('Fréquence')
    plt.grid(True)
    plt.show()
```

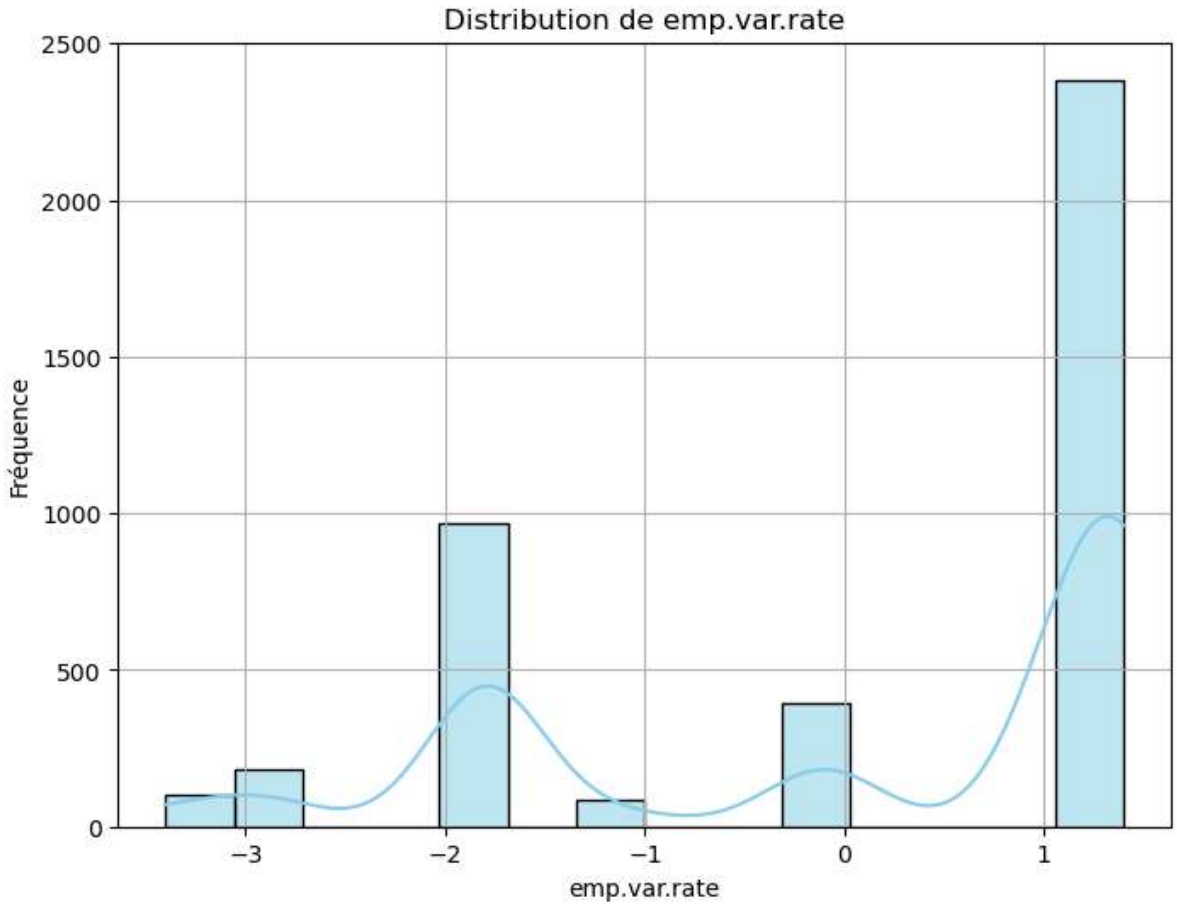

Distribution de age
Fréquence
age
0
50
100
150
200
250
300
350
400
20
30
40
50
60
70
80
90

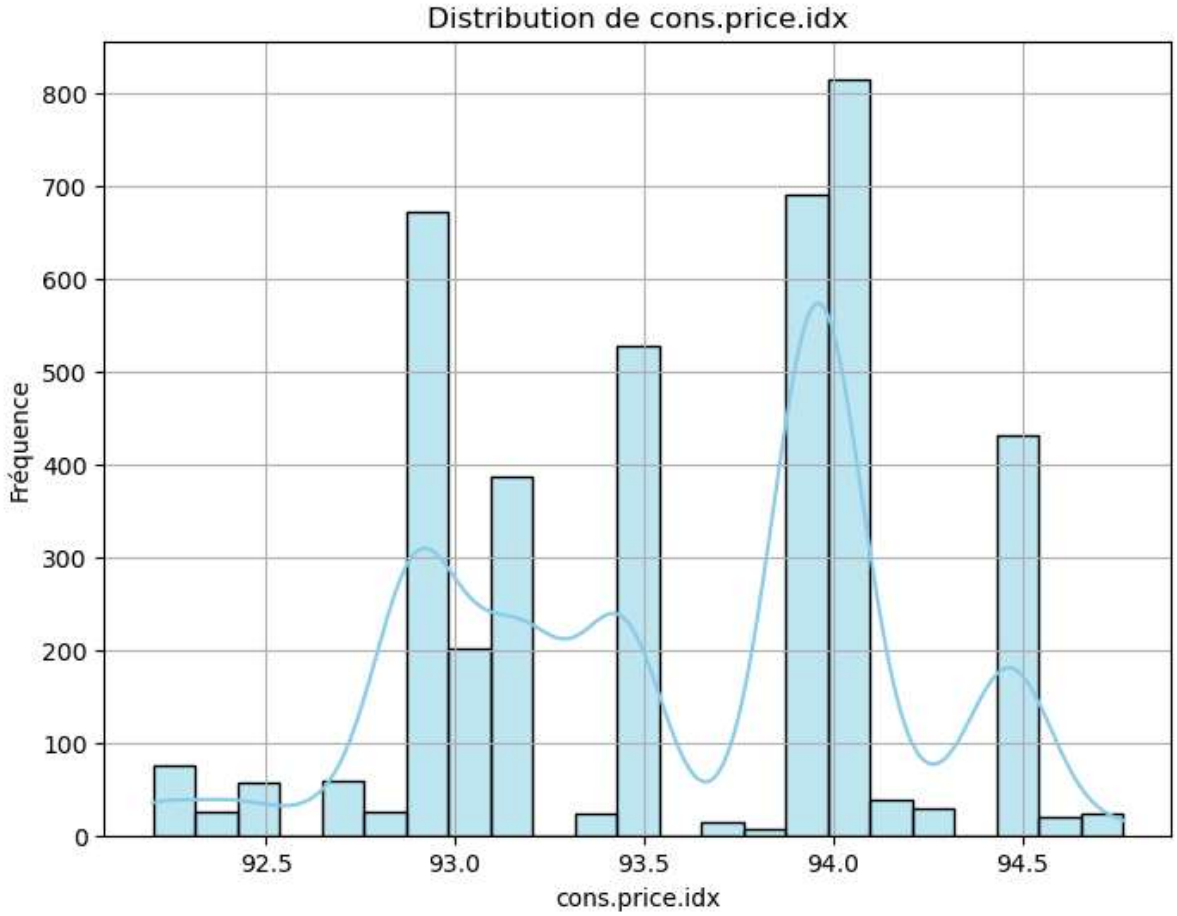

Distribution de duration
Fréquence
duration
0
50
100
150
200
250
300
350
400
0
500
1000
1500
2000
2500
3000
3500

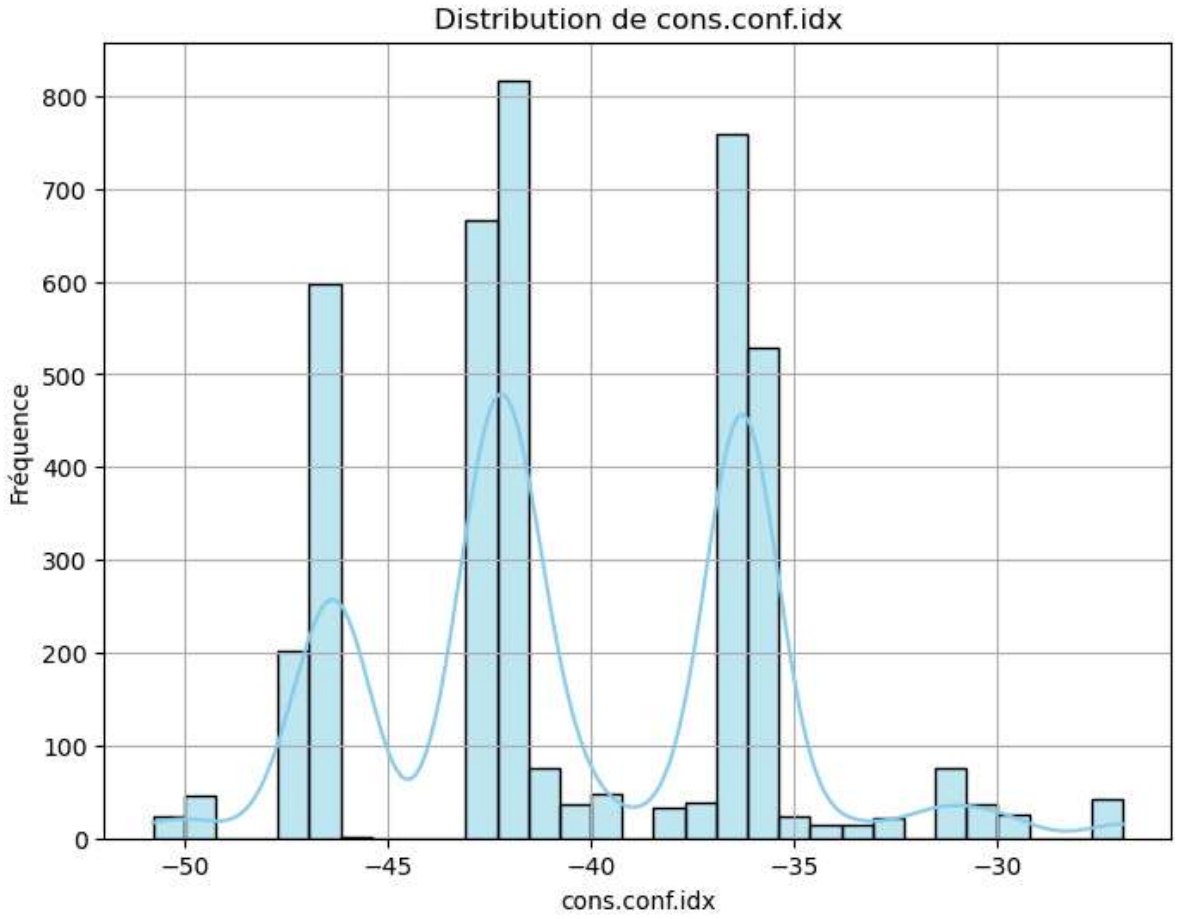

Distribution de campaign
Fréquence
campaign
0
250
500
750
1000
1250
1500
1750
0
5
10
15
20
25
30
35

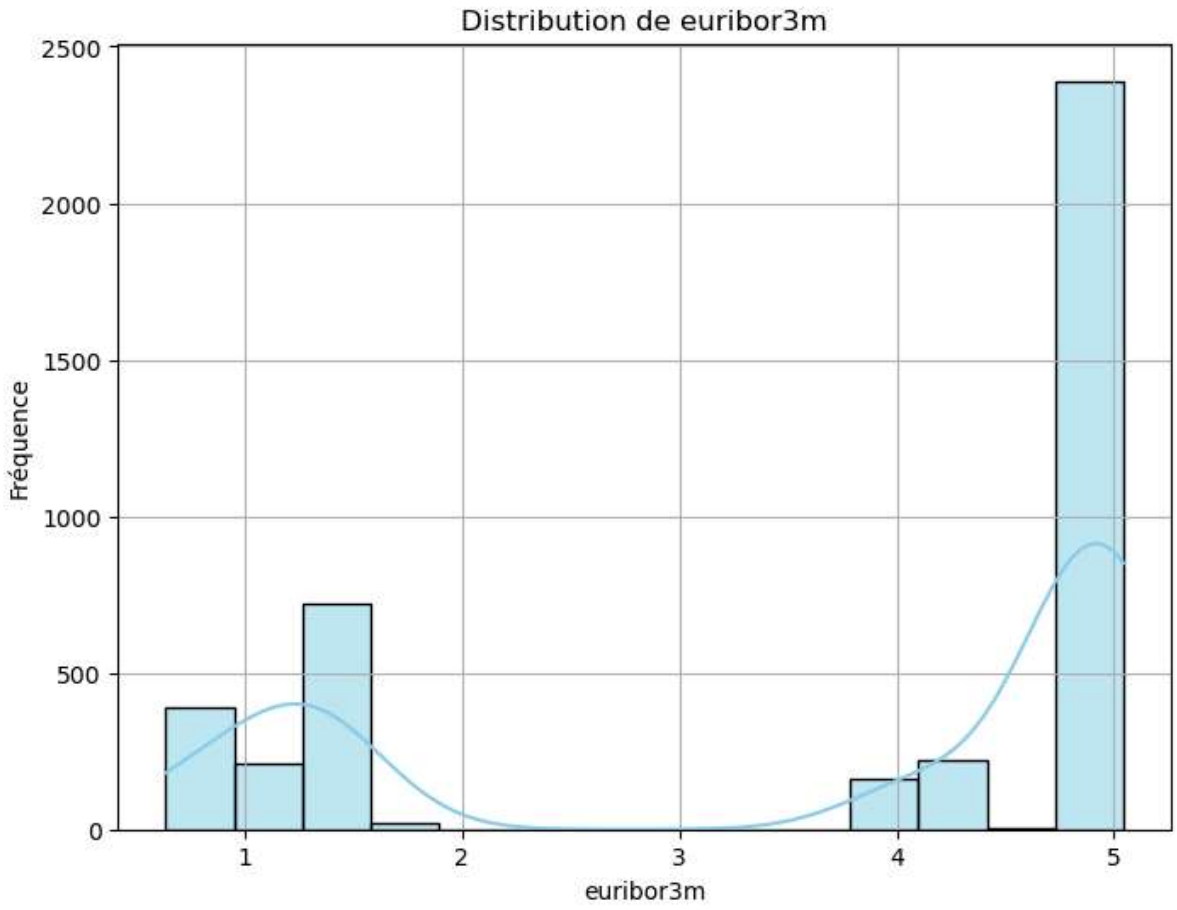

Distribution de pdays
Fréquence
pdays
0
500
1000
1500
2000
2500
3000
3500
4000
0
200
400
600
800
1000

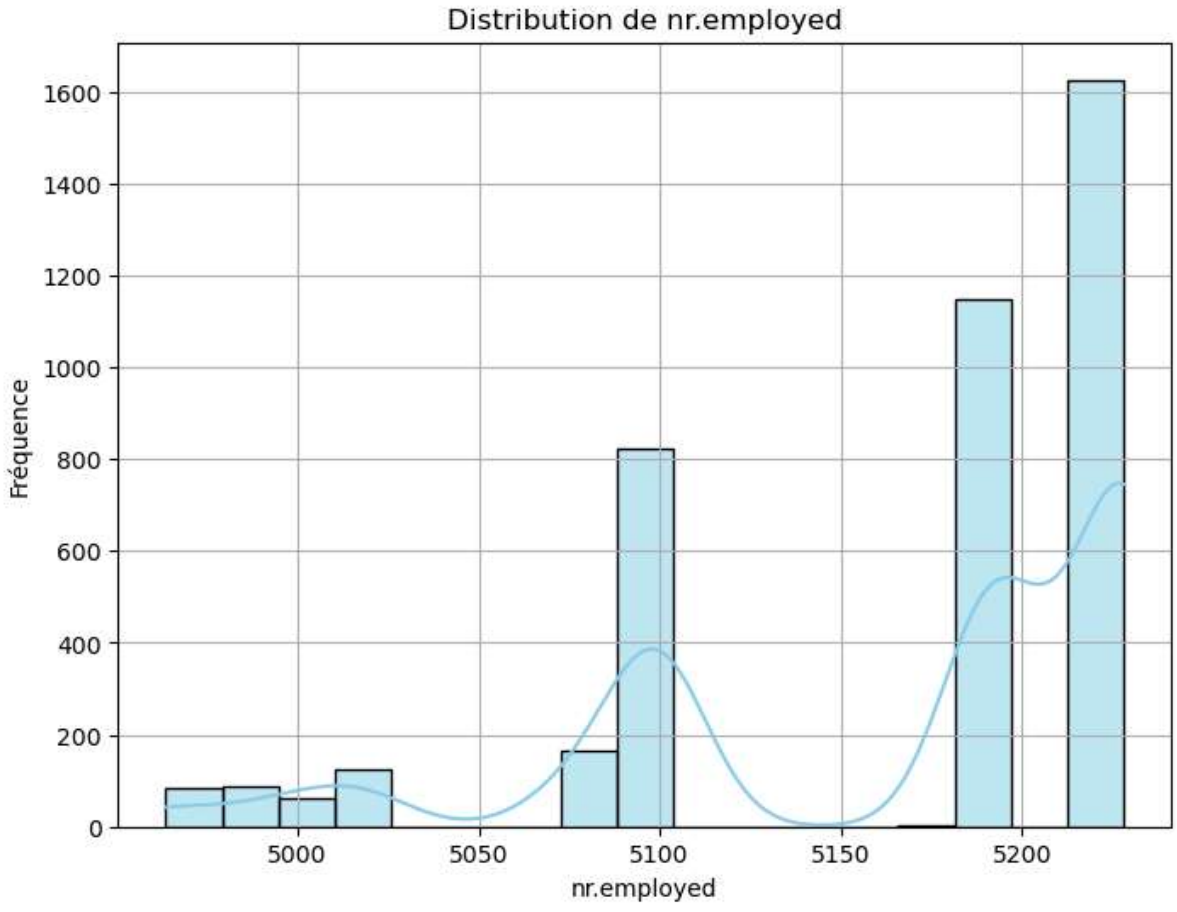

Distribution de previous
Fréquence
previous
0
1000
2000
3000
4000
5000
6000
0
1
2
3
4
5
6


Distribution de emp.var.rate
Fréquence
emp.var.rate
0
500
1000
1500
2000
2500
−3
−2
−1
0
1


Distribution de cons.price.idx
Fréquence
cons.price.idx
0
100
200
300
400
500
600
700
800
92.5
93.0
93.5
94.0
94.5

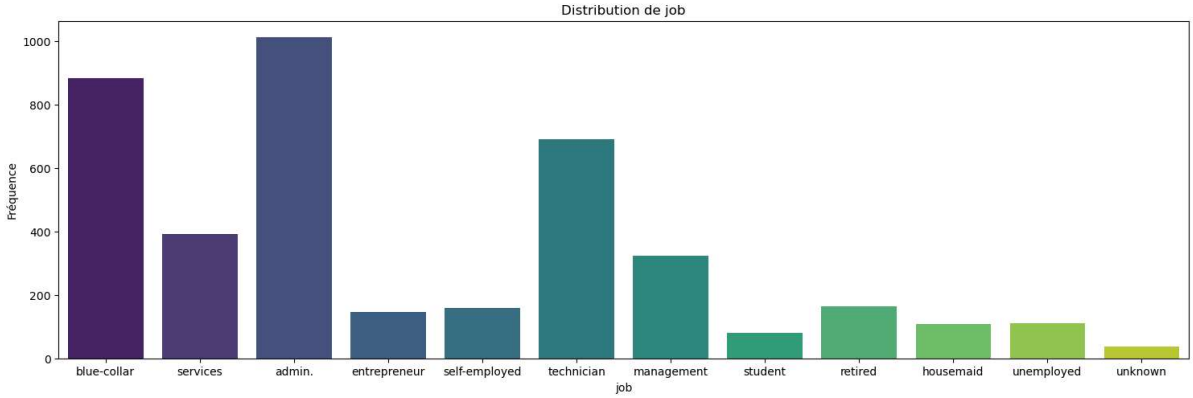

Distribution de cons.conf.idx
Fréquence
cons.conf.idx
0
100
200
300
400
500
600
700
800
−50
−45
−40
−35
−30

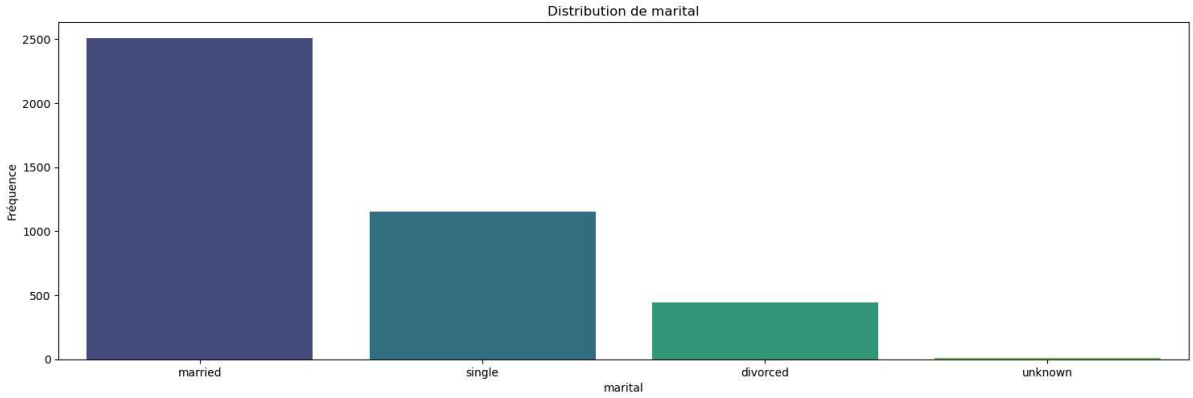


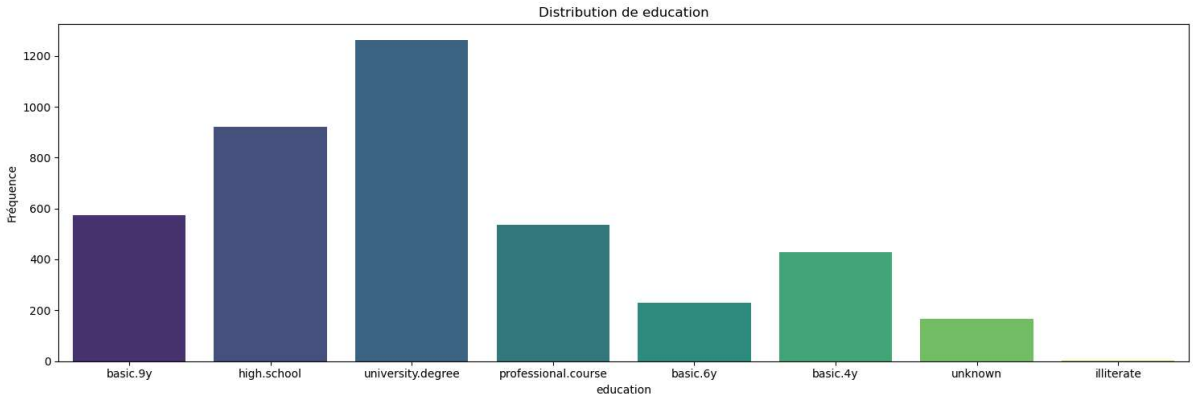


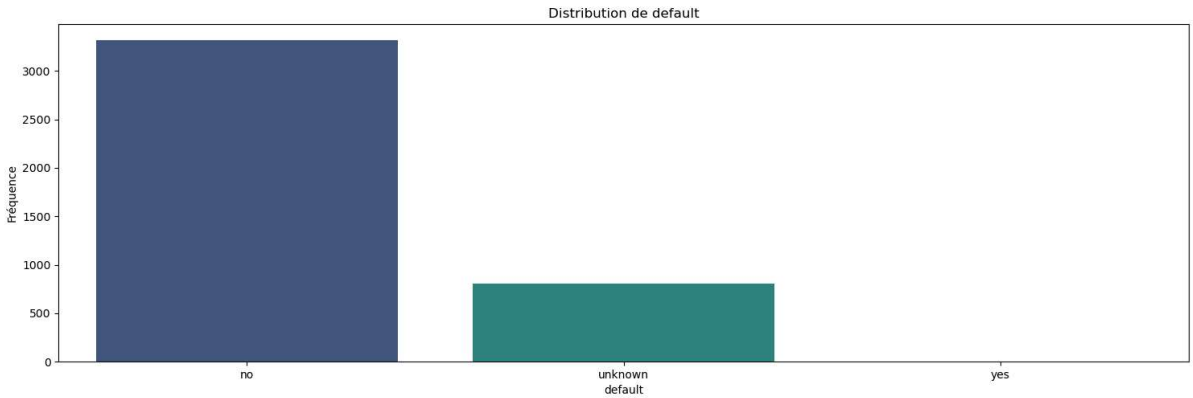
In [12]:
```
num_plots = len(df_categorical.columns)
plt.figure(figsize=(15, 5*num_plots))

for i, column in enumerate(df_categorical.columns, 1):
    plt.subplot(num_plots, 1, i)
    sns.countplot(data=df_categorical, x=column, palette='viridis')
```

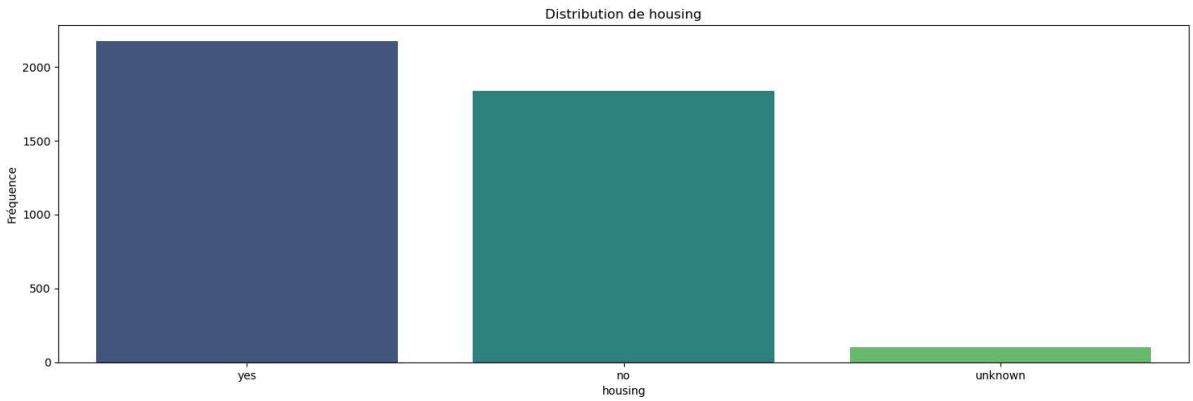
```
    plt.title(f'Distribution de {column}')
    plt.xlabel(column)
    plt.ylabel('Fréquence')

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

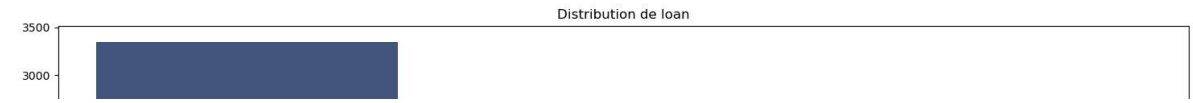

Distribution de job
Fréquence
job
blue-collar
services
admin.
entrepreneur
self-employed
technician
management
student
retired
housemaid
unemployed
unknown

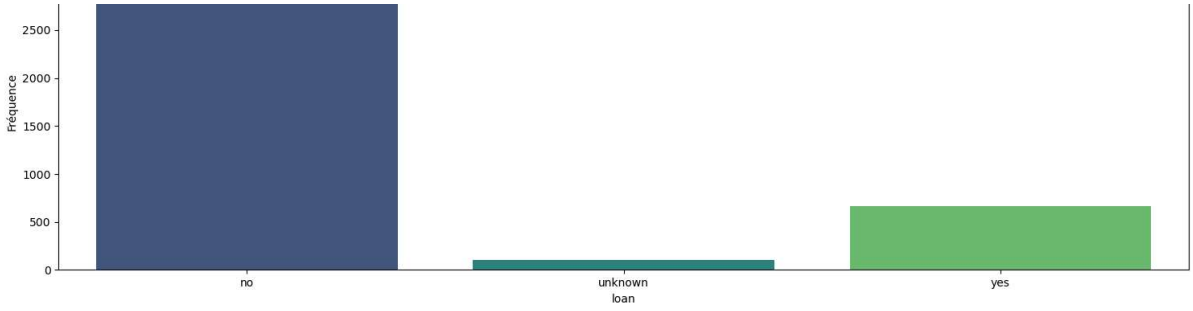

Distribution de marital
Fréquence
marital
married
single
divorced
unknown

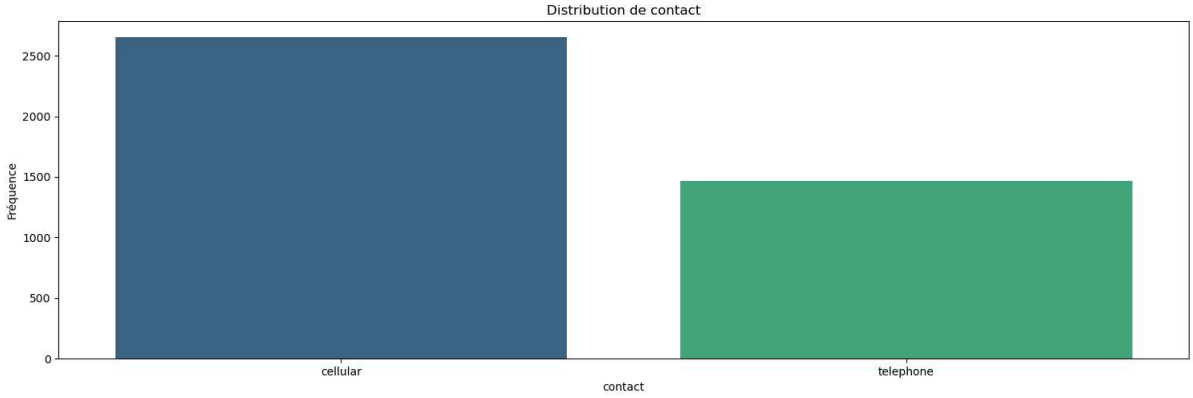

Distribution de education
Fréquence
education
basic.9y
high.school
university.degree
professional.course
basic.6y
basic.4y
unknown
illiterate

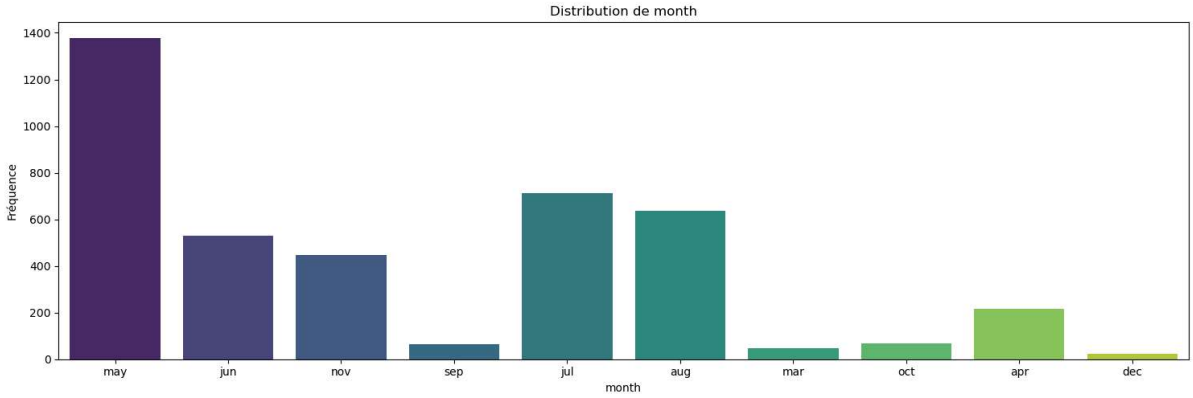

Distribution de default
Fréquence
default
no
unknown
yes

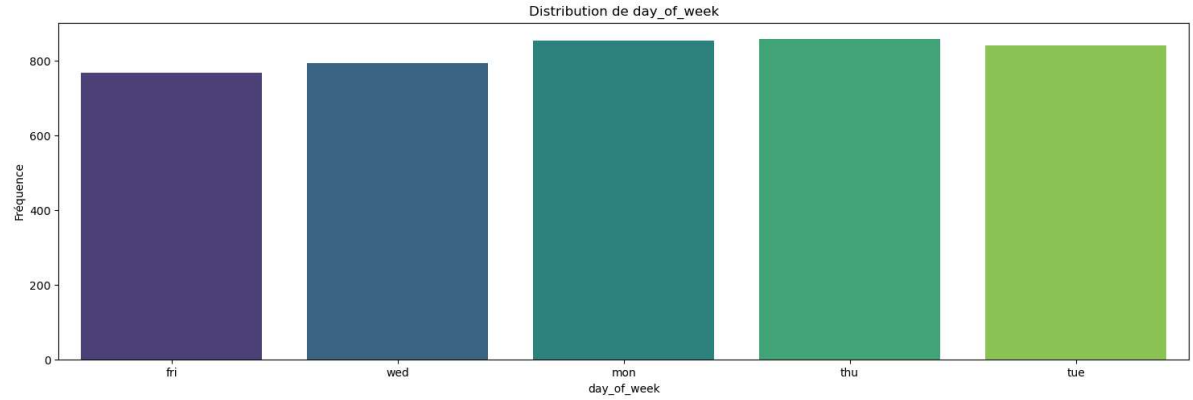

Distribution de housing
Fréquence
housing
yes
no
unknown

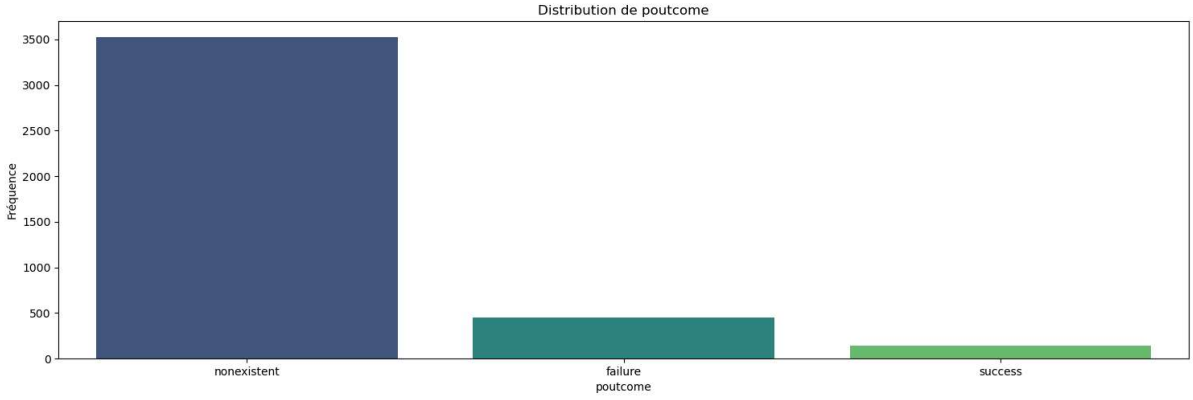

Distribution de loan

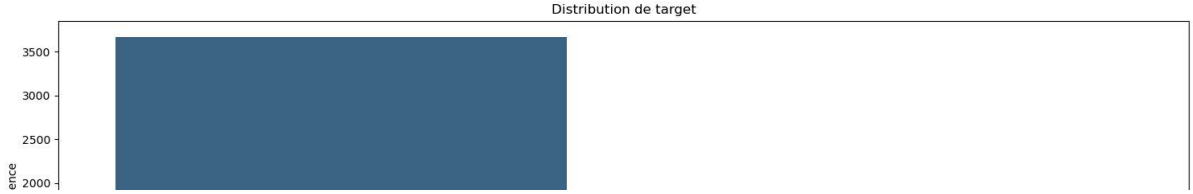

Fréquence
2500
2000
1500
1000
500
0
no
unknown
yes
loan

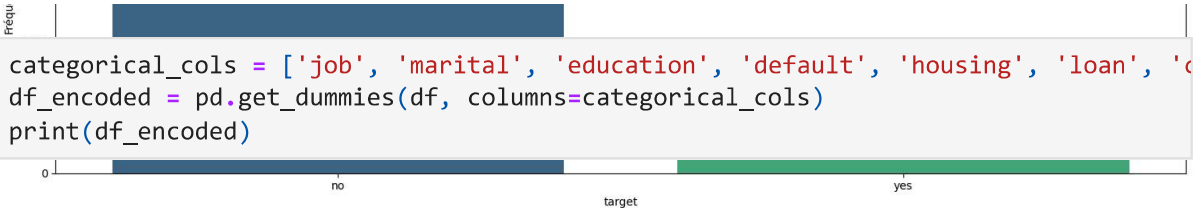

Distribution de contact
Fréquence
2500
2000
1500
1000
500
0
cellular
telephone
contact


Distribution de month
Fréquence
1400
1200
1000
800
600
400
200
0
may
jun
nov
sep
jul
aug
mar
oct
apr
dec
month


Distribution de day_of_week
Fréquence
800
600
400
200
0
fri
wed
mon
thu
tue
day_of_week


Distribution de poutcome
Fréquence
3500
3000
2500
2000
1500
1000
500
0
nonexistent
failure
success
poutcome


Distribution de target
3500
3000
2500
2000



In [13]:
```
categorical_cols = ['job', 'marital', 'education', 'default', 'housing', 'loan', 'c
df_encoded = pd.get_dummies(df, columns=categorical_cols)
print(df_encoded)
```

```
      age  duration  campaign  pdays  previous  emp.var.rate  cons.price.idx  \
0      30       487         2    999         0          -1.8          92.893
1      39       346         4    999         0           1.1          93.994
2      25       227         1    999         0           1.4          94.465
3      38        17         3    999         0           1.4          94.465
4      47        58         1    999         0          -0.1          93.200
...   ...       ...       ...    ...       ...           ...             ...
4114   30        53         1    999         0           1.4          93.918
4115   39       219         1    999         0           1.4          93.918
4116   27        64         2    999         1          -1.8          92.893
4117   58       528         1    999         0           1.4          93.444
4118   34       175         1    999         0          -0.1          93.200

      cons.conf.idx  euribor3m  nr.employed  ... month_oct  month_sep  \
0             -46.2      1.313       5099.1  ...     False      False
1             -36.4      4.855       5191.0  ...     False      False
2             -41.8      4.962       5228.1  ...     False      False
3             -41.8      4.959       5228.1  ...     False      False
4             -42.0      4.191       5195.8  ...     False      False
...             ...        ...          ...  ...       ...        ...
4114          -42.7      4.958       5228.1  ...     False      False
4115          -42.7      4.959       5228.1  ...     False      False
4116          -46.2      1.354       5099.1  ...     False      False
4117          -36.1      4.966       5228.1  ...     False      False
4118          -42.0      4.120       5195.8  ...     False      False

      day_of_week_fri  day_of_week_mon  day_of_week_thu  day_of_week_tue  \
0                True            False            False            False
1                True            False            False            False
2               False            False            False            False
3                True            False            False            False
4               False             True            False            False
...               ...              ...              ...              ...
4114            False            False             True            False
4115             True            False            False            False
4116            False             True            False            False
4117             True            False            False            False
4118            False            False            False            False

      day_of_week_wed  poutcome_failure  poutcome_nonexistent  \
0               False             False                  True
1               False             False                  True
2                True             False                  True
3               False             False                  True
4               False             False                  True
...               ...               ...                   ...
4114            False             False                  True
4115            False             False                  True
4116            False              True                 False
4117            False             False                  True
4118             True             False                  True

      poutcome_success
0                False
1                False
2                False
3                False
4                False
...                ...
4114             False
4115             False
4116             False
4117             False
4118             False
```

```
[4119 rows x 64 columns]
```

In [14]:
```
df_encoded['target'].value_counts()
```

Out[14]:
```
target
no     3668
yes     451
Name: count, dtype: int64
```

In [27]:
```
X = df_encoded.drop('target',axis=1)
Y = df_encoded['target']
print(x.shape)
print(y.shape)
print(type(x))
print(type(y))
```

```
(4119, 63)
(4119,)
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'>
```

In [28]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_st

print("Taille de l'ensemble d'entraînement:", len(X_train))
print("Taille de l'ensemble de test:", len(X_test))
```

```
Taille de l'ensemble d'entraînement: 3089
Taille de l'ensemble de test: 1030
```

In [29]:
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_st
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

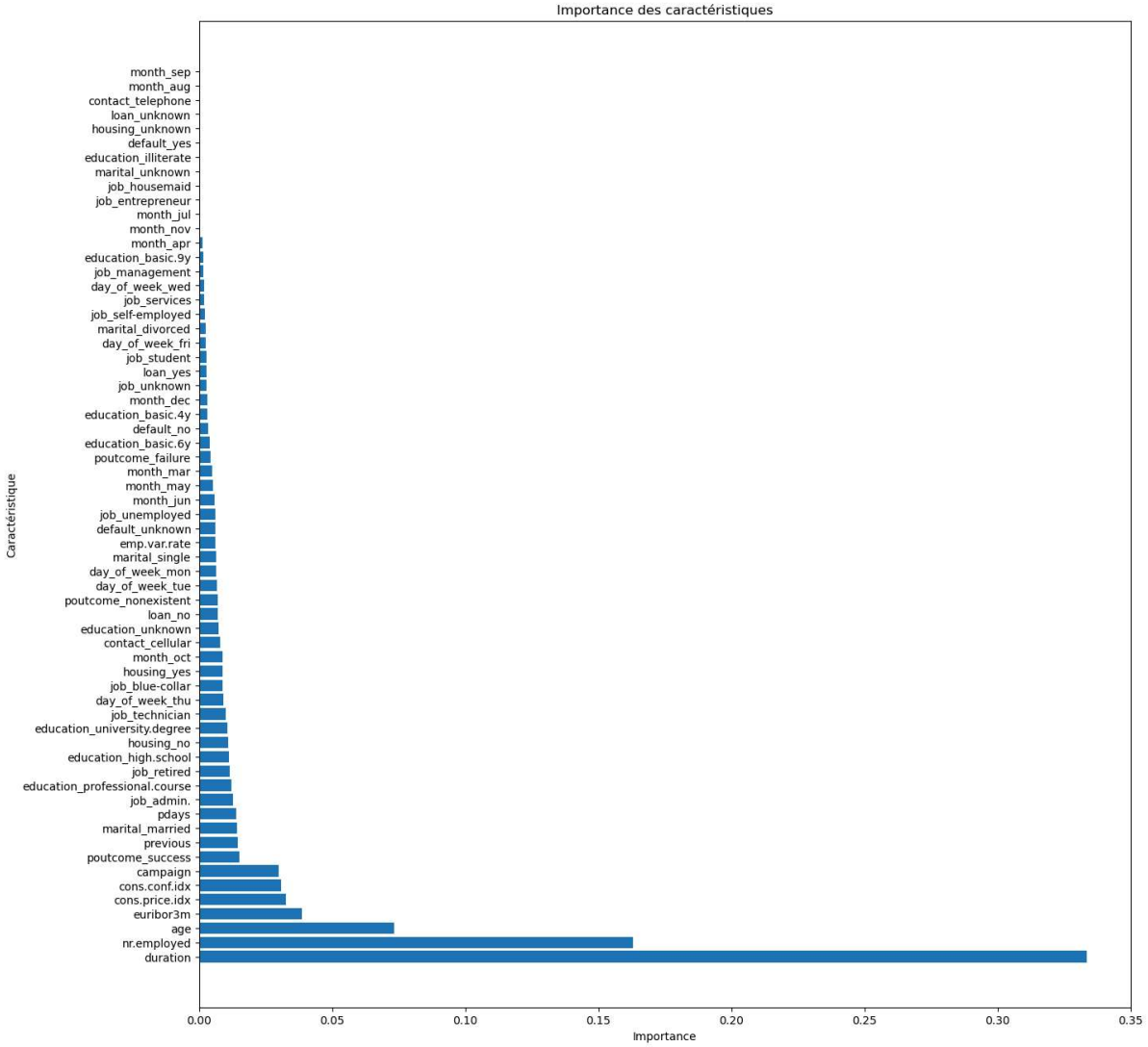
In [30]:
```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8883495145631068
```

In [32]:
```
feature_importances = clf.feature_importances_
feature_importance_dict = dict(zip(X.columns, feature_importances))
sorted_features = sorted(feature_importance_dict.items(), key=lambda x: x[1], rever

print("Caractéristiques importantes :")
for feature, importance in sorted_features:
    print(f"{feature}: {importance}")
```

```
Caractéristiques importantes :
duration: 0.33350863560236477
nr.employed: 0.16315978322217245
age: 0.07332839905625593
euribor3m: 0.03867597955363051
cons.price.idx: 0.03249293067723582
cons.conf.idx: 0.030890977085346515
campaign: 0.02980647689928591
poutcome_success: 0.01502593277824243
previous: 0.014644049611674412
marital_married: 0.014174976872401178
pdays: 0.01396524260608575
job_admin.: 0.012838091360922113
education_professional.course: 0.012102253882318531
job_retired: 0.011570746019616454
education_high.school: 0.011194535683212329
housing_no: 0.010960033664227714
education_university.degree: 0.010798341307117376
job_technician: 0.010188910767095456
day_of_week_thu: 0.009198599682476832
job_blue-collar: 0.008975934309572824
housing_yes: 0.008829274178329329
month_oct: 0.008809901145028035
contact_cellular: 0.007909417006674451
education_unknown: 0.007403281325364322
loan_no: 0.007040363819696394
poutcome_nonexistent: 0.006928306994886804
day_of_week_tue: 0.006850931341941681
day_of_week_mon: 0.006320439581576696
marital_single: 0.006302019356930936
emp.var.rate: 0.006246493067232722
default_unknown: 0.0060646361393186295
job_unemployed: 0.0060425999344159865
month_jun: 0.0057390721732683316
month_may: 0.005337059871276105
month_mar: 0.004995978862559062
poutcome_failure: 0.0043840134656910855
education_basic.6y: 0.0039273453963482655
default_no: 0.0032880100992683146
education_basic.4y: 0.0031514241335199957
month_dec: 0.003082509468064045
job_unknown: 0.002882965376894681
loan_yes: 0.002846761990708497
job_student: 0.0028182943708014133
day_of_week_fri: 0.0026304080794146514
marital_divorced: 0.0023693013950609916
job_self-employed: 0.0021048246468800937
job_services: 0.0018761113539083486
day_of_week_wed: 0.0018527856626935355
job_management: 0.0014796045446707406
education_basic.9y: 0.0014796045446707406
month_apr: 0.001196786608466346
month_nov: 0.0002463705046370558
month_jul: 6.227291854675453e-05
job_entrepreneur: 0.0
job_housemaid: 0.0
marital_unknown: 0.0
education_illiterate: 0.0
default_yes: 0.0
housing_unknown: 0.0
loan_unknown: 0.0
contact_telephone: 0.0
month_aug: 0.0
month_sep: 0.0
```

In [33]:
```
plt.figure(figsize=(15,16))
plt.barh(range(len(sorted_features)), [val[1] for val in sorted_features], align='c
plt.yticks(range(len(sorted_features)), [val[0] for val in sorted_features])
plt.xlabel('Importance')
plt.ylabel('Caractéristique')
plt.title('Importance des caractéristiques')
plt.show()
```



In [35]:
```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

clf = DecisionTreeClassifier()

grid_search = GridSearchCV(estimator=clf, param_grid=param_grid, cv=5)

grid_search.fit(X_train, y_train)

print("Meilleurs hyperparamètres :", grid_search.best_params_)

best_clf = grid_search.best_estimator_

y_pred = best_clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy du meilleur modèle:", accuracy)
```

```
Meilleurs hyperparamètres : {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_
split': 5}
Accuracy du meilleur modèle: 0.9029126213592233
```

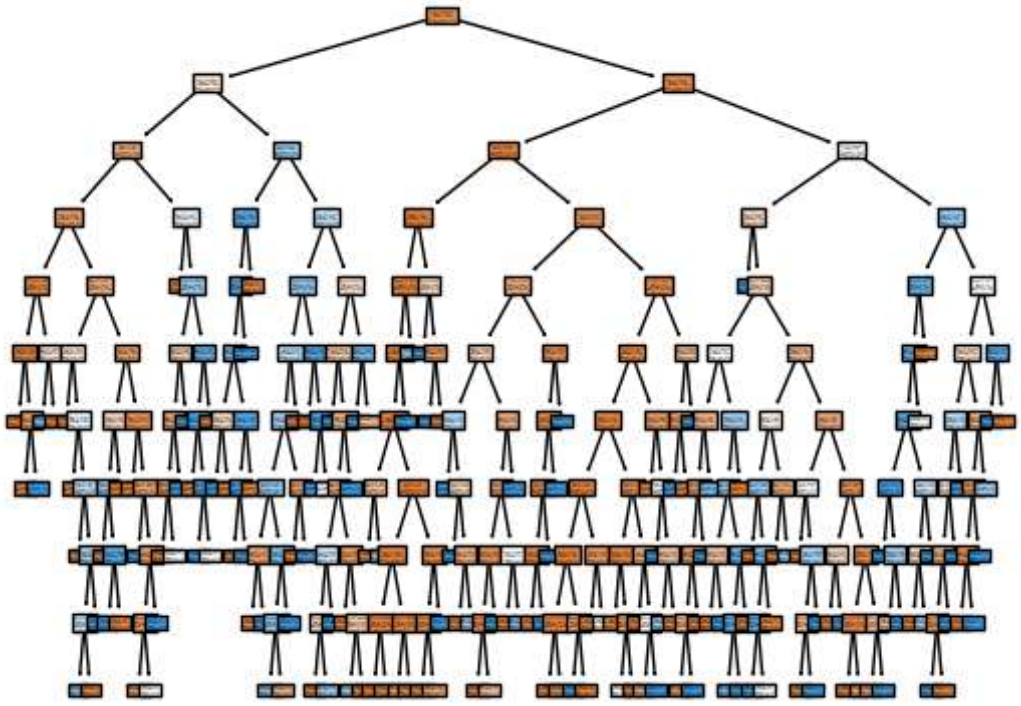
In [39]:
```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(criterion='gini',max_depth=10,min_samples_split=5)
dt.fit(X_train,y_train)
```

Out[39]:
```
DecisionTreeClassifier(max_depth=10, min_samples_split=5)
```

In [45]:
```
y_pred = dt.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy du modèle sur les données de test:", accuracy)
```

```
Accuracy du modèle sur les données de test: 0.9
```

In [46]:
```
plot_tree(dt,class_names=cn,filled=True)
plt.show()
```



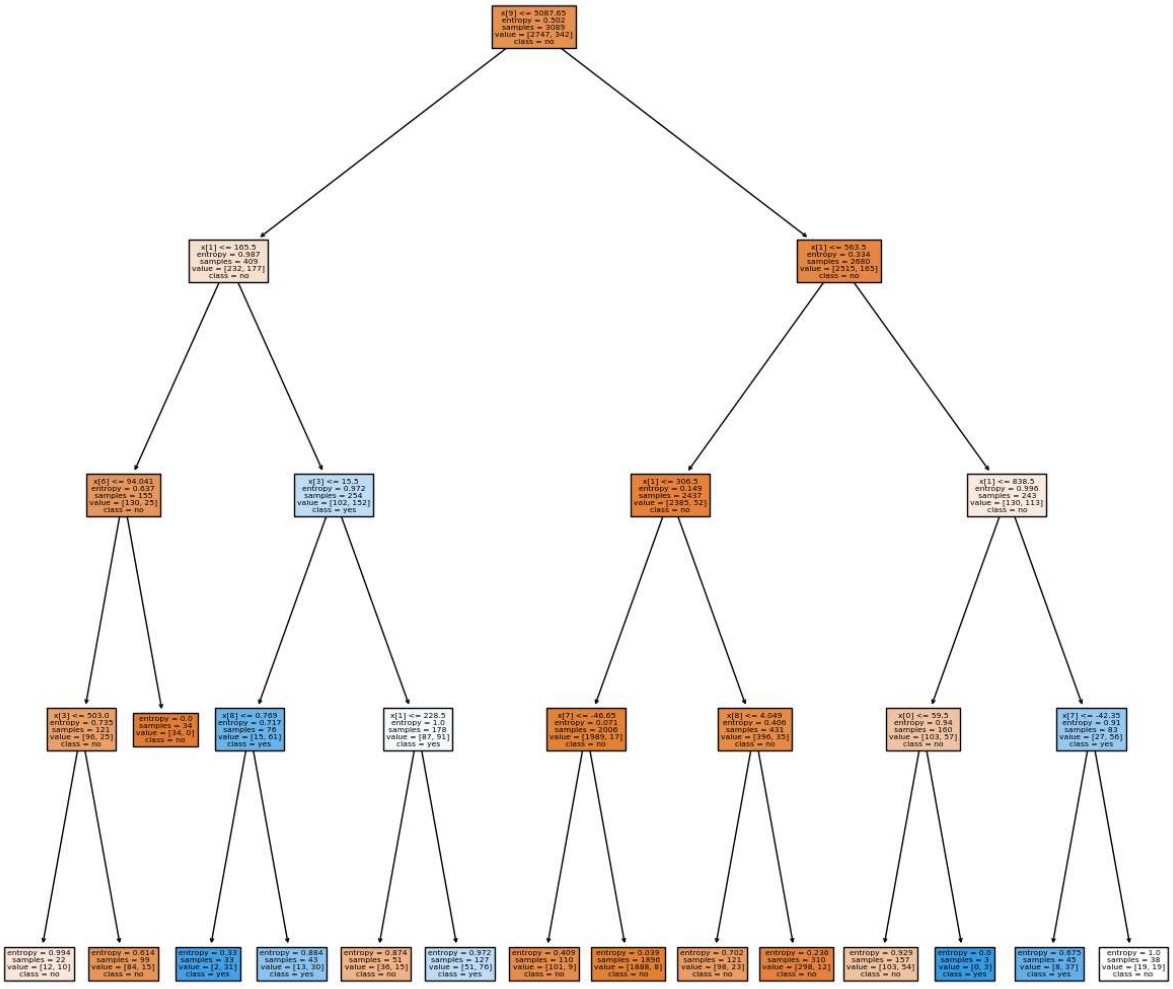
In [50]:
```
 dt1 =DecisionTreeClassifier(criterion='entropy',max_depth=4,min_samples_split=15)
dt1.fit(X_train,y_train)
```

Out[50]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=4, min_samples_spli
t=15)
```

In [52]:
```
from sklearn.metrics import accuracy_score
y_pred = dt.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy du modèle sur les données de test:", accuracy)
```

```
Accuracy du modèle sur les données de test: 0.9
```

In [54]:
```
plt.figure(figsize=(15,15))
plot_tree(dt1,class_names=cn,filled=True)
plt.show()
```



In [ ]: