BANK MARKETING

Data Set Information:-

This data set contains records relevant to a direct marketing campaign of a Portuguese banking institution. The marketing campaign was executed through phone calls. Often, more than one call needs to be made to a single client before they either decline or agree to a term deposit subscription. The classification goal is to predict if the client will subscribe (yes/no) to the term deposit (variable y).

This is a modified version of the classic bank marketing data set originally shared in the UCI Machine Learning Repository. There are four datasets available on UCI's repository: 1) 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] 2) bank-additional.csv with 10% of the examples (4119), randomly selected from 1), and 20 inputs. 3) bank-full.csv with all examples and 17 inputs, ordered by date (older version of this data set with less inputs). 4) bank.csv with 10% of the examples and 17 inputs, randomly selected from 3 (older version of this data set with less inputs). Note: The smallest datasets are provided to test more computationally demanding machine learning algorithms (e.g., SVM).

This data set is a copy of data set no. 1 (bank-additional-full.csv) from the list above with one input feature (representing duration of phone call) removed. The following is a note from the variable description in the original data set:

duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should

only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

Contents

Input variables:-

Bank Client Data:

1 - age (numeric)

2 - job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')

3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)

4 - education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')

5 - default: has credit in default? (categorical: 'no','yes','unknown')

6 - housing: has housing loan? (categorical: 'no','yes','unknown')

7 - loan: has personal loan? (categorical: 'no','yes','unknown')

related with the last contact of the current campaign:

8 - contact: contact communication type (categorical: 'cellular','telephone')

9 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')

10 - day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')

Other Attributes:

11 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)

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

13 - previous: number of contacts performed before this campaign and for this client (numeric)

14 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')

Social and Economic Context Attributes:

15 - emp.var.rate: employment variation rate - quarterly indicator (numeric)

16 - cons.price.idx: consumer price index - monthly indicator (numeric)

17 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)

18 - euribor3m: euribor 3 month rate - daily indicator (numeric)

19 - nr.employed: number of employees - quarterly indicator (numeric)

Output variable (desired target):

20 - y - has the client subscribed a term deposit? (binary: 'yes','no')

Context:-

Use this data set to test the performance of your classification models and to explore the best strategies to improve a banking institution's next direct marketing campaign.

Term deposits are cash investment held at a financial institution and are a major source of revenue for banks--making them important for financial institutions to market. Telemarketing remains to be a popular marketing technique because of the potential effectiveness of human-to-human contact provided by a telephone call, which is sometimes quite the opposite of many impersonal and robotic marketing messages relayed through social and digital media. However, executing such direct marketing effort usually requires a huge investment by the business as large call centers need to be contracted to contact clients directly.

Import the libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Load the dataset

```
df=pd.read_csv('/content/bank-direct-marketing-campaigns.csv')
df
```

	age	job	marital	education	default	housing	loan	contact	m
0	56	housemaid	married	basic.4y	no	no	no	telephone	
1	57	services	married	high.school	unknown	no	no	telephone	
2	37	services	married	high.school	no	yes	no	telephone	
3	40	admin.	married	basic.6y	no	no	no	telephone	
...	...	...	...	...	...	...	...	...	

Data Exploration

```
df.head()
```

	age	job	marital	education	default	housing	loan	contact	month	day_o
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	
1	57	services	married	high.school	unknown	no	no	telephone	may	
2	37	services	married	high.school	no	yes	no	telephone	may	
3	40	admin.	married	basic.6y	no	no	no	telephone	may	
4	56	services	married	high.school	no	no	yes	telephone	may	

```
df.tail()
```

	age	job	marital	education	default	housing	loan	contact	mon
41183	73	retired	married	professional.course	no	yes	no	cellular	n
41184	46	blue-collar	married	professional.course	no	no	no	cellular	n
41185	56	retired	married	university.degree	no	yes	no	cellular	n
41186	44	technician	married	professional.course	no	no	no	cellular	n
41187	74	retired	married	professional.course	no	yes	no	cellular	n

```
df.shape
```

```
(41188, 20)
```

```
df.dtypes
```

```
age               int64
job              object
marital          object
education        object
default          object
housing          object
loan             object
contact          object
```

```
month             object
day_of_week       object
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
y                 object
dtype: object
```

```
df.columns
```

```
Index(['age', 'job', 'marital', 'education', 'default', 'housing', 'loan',
       'contact', 'month', 'day_of_week', 'campaign', 'pdays', 'previous',
       'poutcome', 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx',
       'euribor3m', 'nr.employed', 'y'],
      dtype='object')
```

```
df.describe()
```

	age	campaign	pdays	previous	emp.var.rate	cons.price.
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000
mean	40.02406	2.567593	962.475454	0.172963	0.081886	93.575
std	10.42125	2.770014	186.910907	0.494901	1.570960	0.578
min	17.00000	1.000000	0.000000	0.000000	-3.400000	92.201
25%	32.00000	1.000000	999.000000	0.000000	-1.800000	93.075
50%	38.00000	2.000000	999.000000	0.000000	1.100000	93.749
75%	47.00000	3.000000	999.000000	0.000000	1.400000	93.994
max	98.00000	56.000000	999.000000	7.000000	1.400000	94.767

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 20 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  object
 2   marital         41188 non-null  object
 3   education       41188 non-null  object
 4   default         41188 non-null  object
 5   housing         41188 non-null  object
 6   loan            41188 non-null  object
 7   contact         41188 non-null  object
 8   month           41188 non-null  object
```

```
 9   day_of_week     41188 non-null  object
 10  campaign        41188 non-null  int64
 11  pdays           41188 non-null  int64
 12  previous        41188 non-null  int64
 13  poutcome        41188 non-null  object
 14  emp.var.rate    41188 non-null  float64
 15  cons.price.idx  41188 non-null  float64
 16  cons.conf.idx   41188 non-null  float64
 17  euribor3m       41188 non-null  float64
 18  nr.employed     41188 non-null  float64
 19  y               41188 non-null  object
dtypes: float64(5), int64(4), object(11)
memory usage: 6.3+ MB
```

```
df.dropna()
```

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

41188 rows × 20 columns

```
df.isnull().sum()
```

```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
campaign          0
pdays             0
previous          0
poutcome          0
```

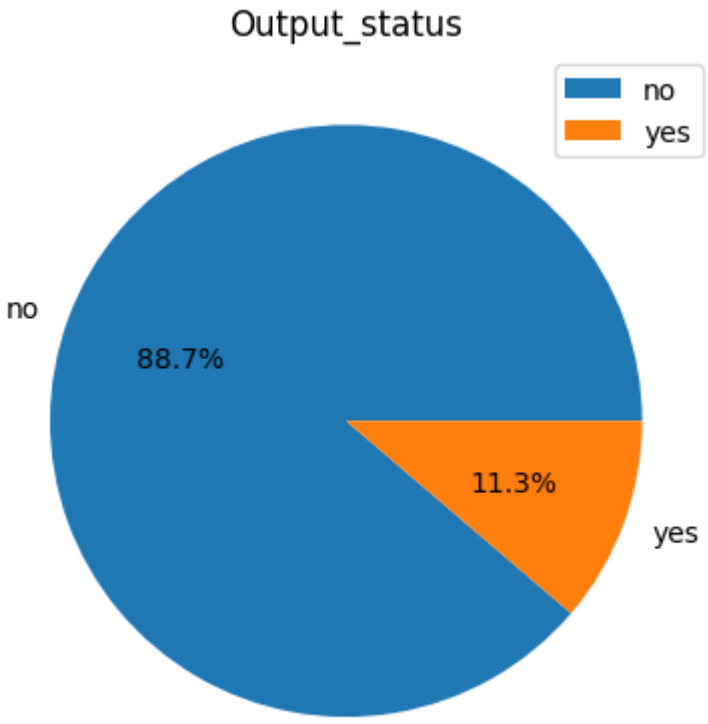
```
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
dtype: int64
```

Data Visualization

```
y_counts=df['y'].value_counts()
y_counts
```

```
no     36548
yes     4640
Name: y, dtype: int64
```

```
mylabels=['no','yes']
plt.pie(y_counts,labels=mylabels,autopct="%1.1f%%")
plt.title('Output_status')
plt.legend(loc='upper right')
plt.show()
```



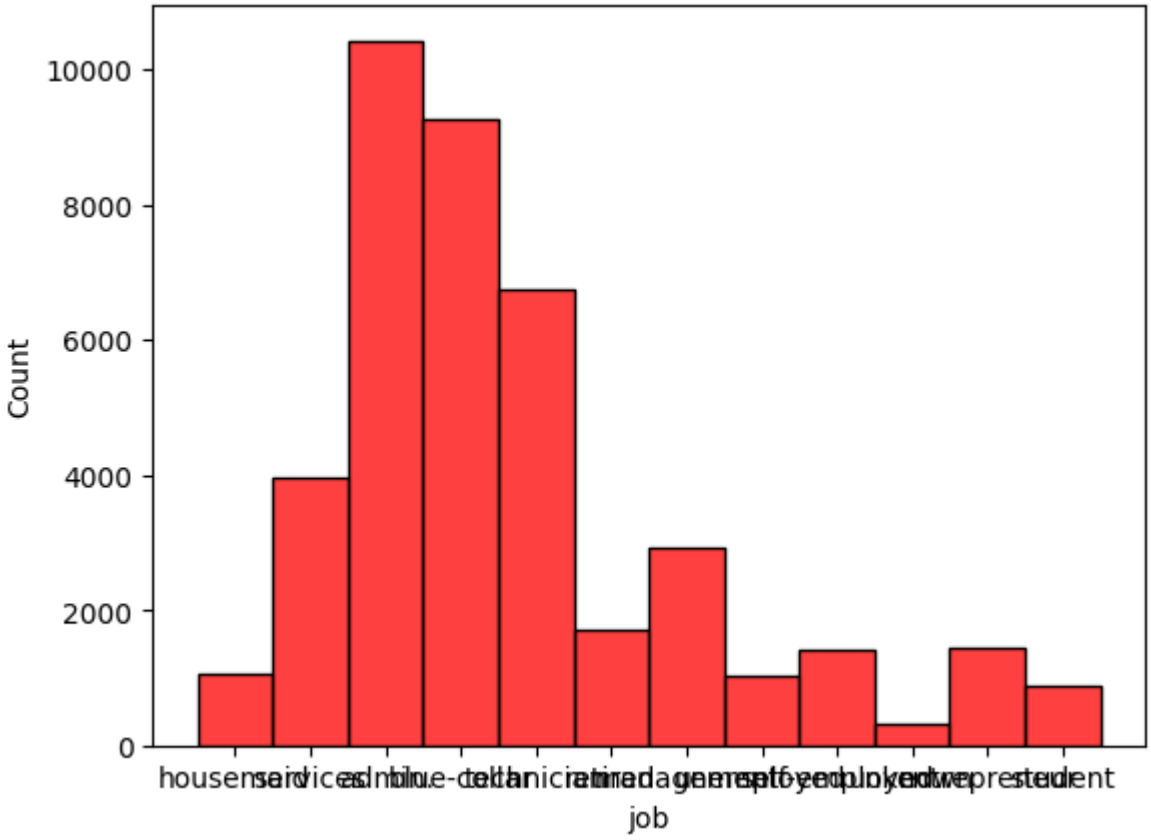

```
df['job'].value_counts()
```

```
admin.          10422
blue-collar      9254
technician       6743
services         3969
management       2924
retired          1720
```

```
entrepreneur      1456
self-employed     1421
housemaid         1060
unemployed        1014
student            875
unknown            330
Name: job, dtype: int64
```

```
sns.histplot(df['job'],color='r')
```

```
<Axes: xlabel='job', ylabel='Count'>
```

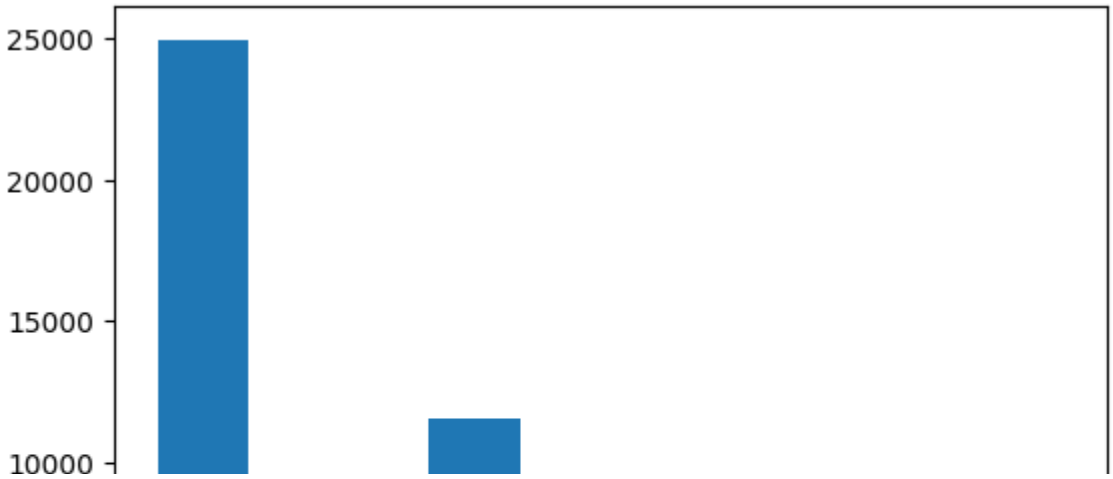



```
df['marital'].value_counts()
```

```
married     24928
single      11568
divorced     4612
unknown        80
Name: marital, dtype: int64
```

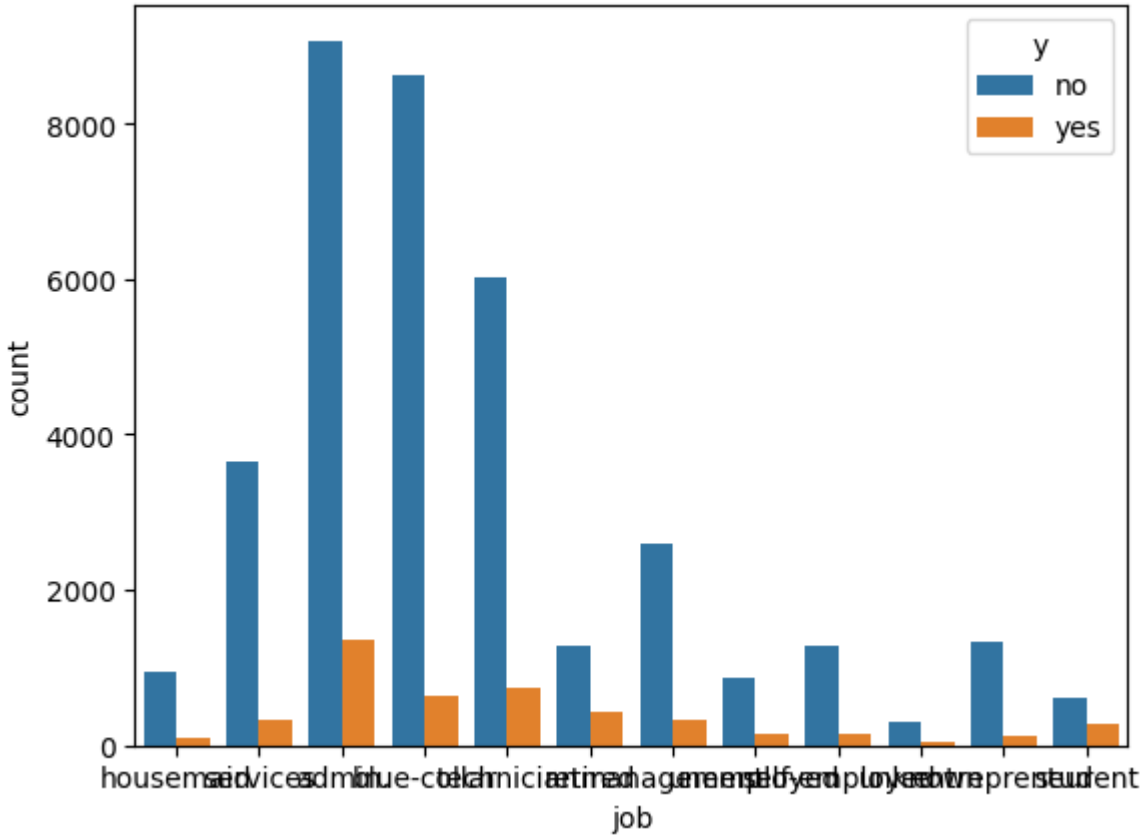
```
plt.hist(df['marital'])
```

```
(array([24928.,     0.,     0., 11568.,     0.,     0.,  4612.,     0.,
            0.,    80.]),
 array([0. , 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.1, 2.4, 2.7, 3. ]),
 <BarContainer object of 10 artists>)
```



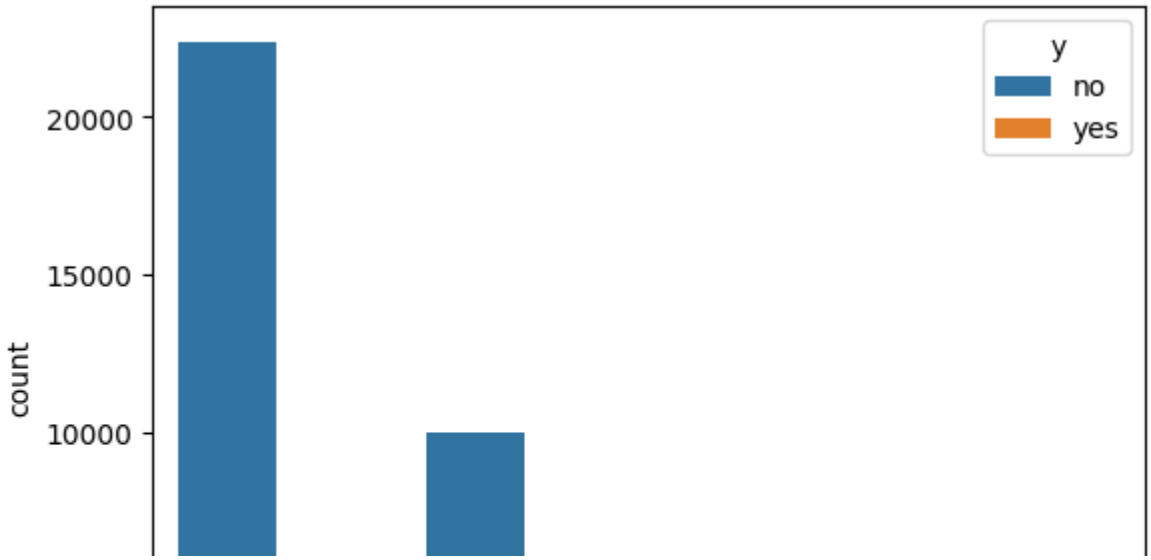

```
sns.countplot(x='job',hue='y',data=df)
```

```
<Axes: xlabel='job', ylabel='count'>
```



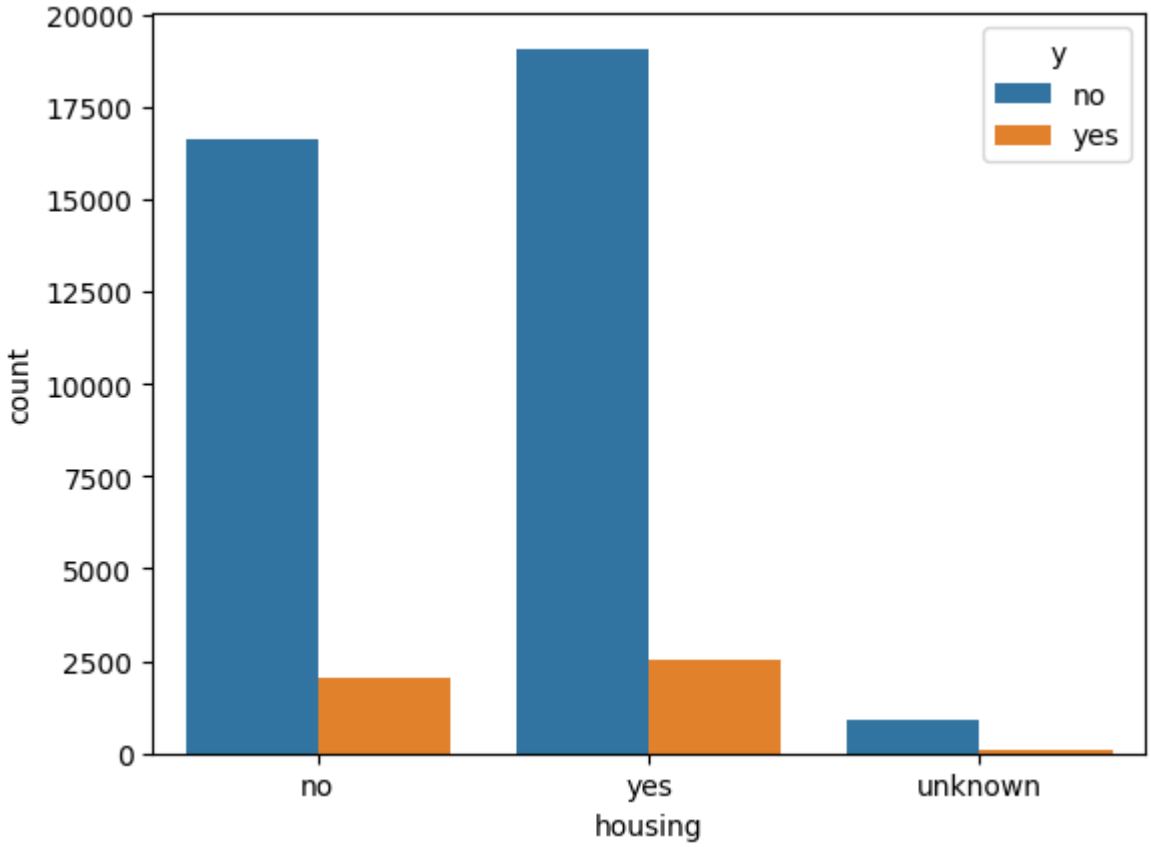

```
sns.countplot(x='marital',hue='y',data=df)
```

```
<Axes: xlabel='marital', ylabel='count'>
```




```
sns.countplot(x='housing',hue='y',data=df)
```

```
<Axes: xlabel='housing', ylabel='count'>
```

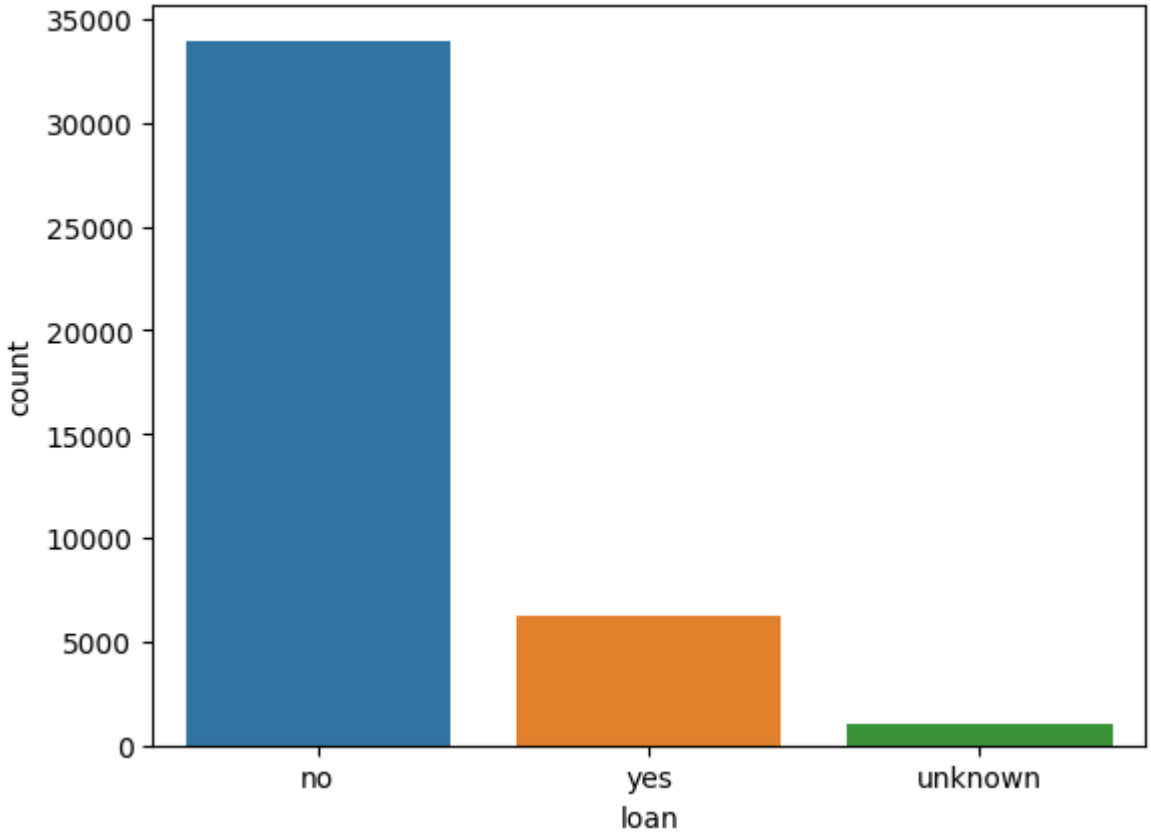



```
sns.countplot(x='default',hue='y',data=df)
```

```
<Axes: xlabel='default', ylabel='count'>
```

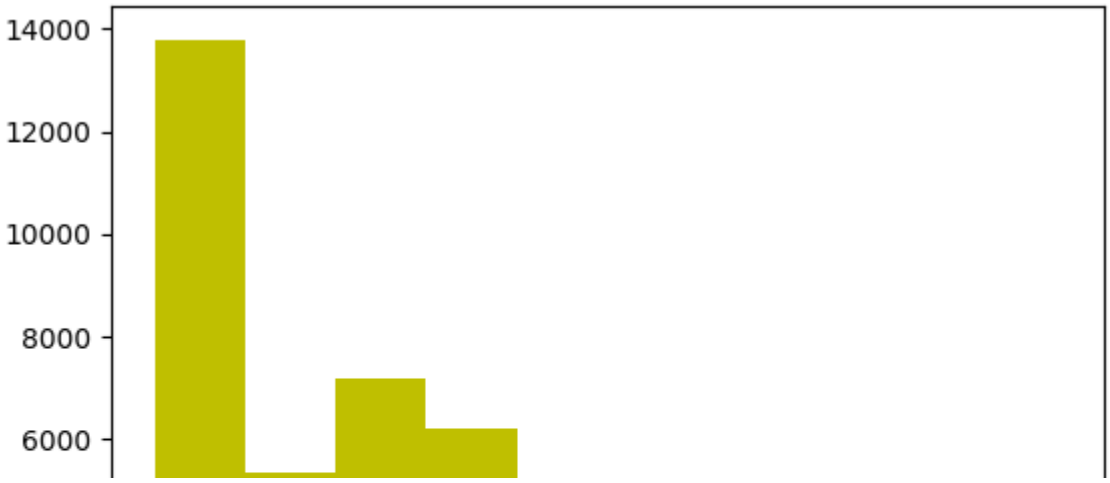

```
sns.countplot(x='loan',data=df)
```

```
<Axes: xlabel='loan', ylabel='count'>
```



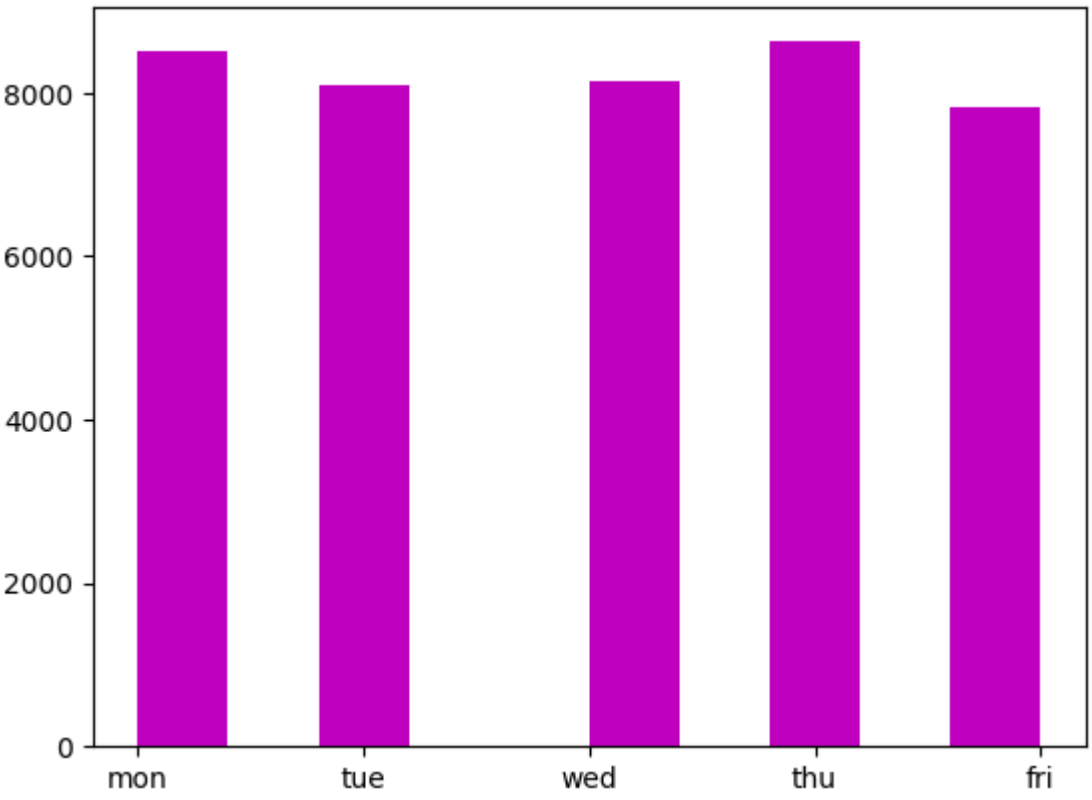

```
plt.hist(df['month'],color='y')
```

```
(array([13769.,  5318.,  7174.,  6178.,   718.,  4101.,   182.,   546.,
         2632.,   570.]),
 array([0. , 0.9, 1.8, 2.7, 3.6, 4.5, 5.4, 6.3, 7.2, 8.1, 9. ]),
 <BarContainer object of 10 artists>)
```



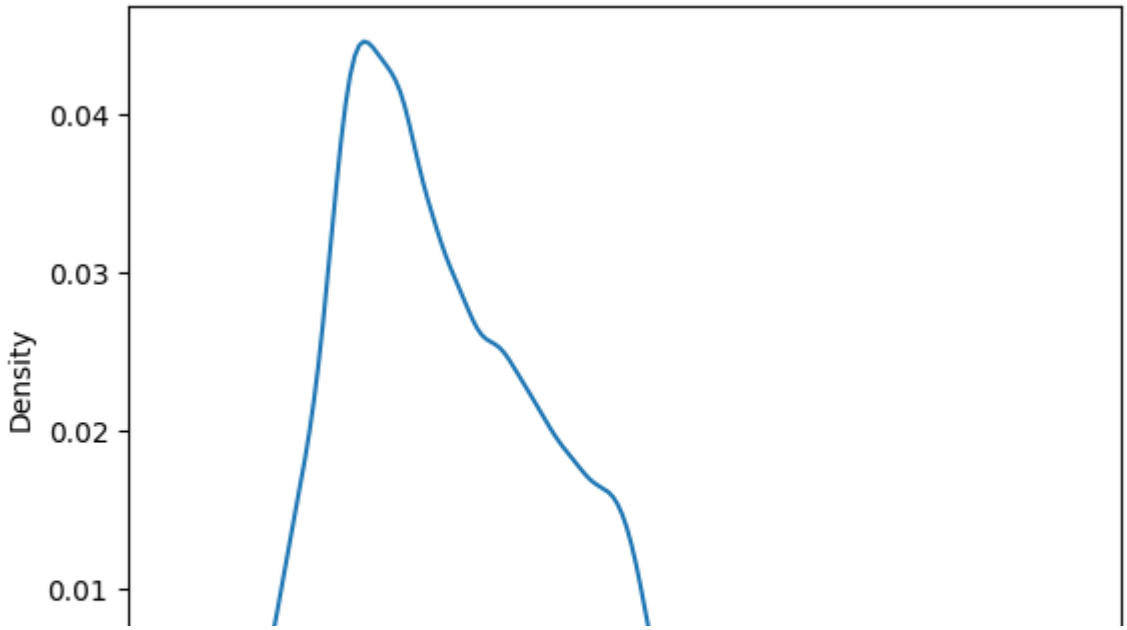

```
plt.hist(df['day_of_week'],color='m')
```

```
(array([8514.,    0., 8090.,    0.,    0., 8134.,    0., 8623.,    0.,
        7827.]),
 array([0. , 0.4, 0.8, 1.2, 1.6, 2. , 2.4, 2.8, 3.2, 3.6, 4. ]),
 <BarContainer object of 10 artists>)
```




```
sns.kdeplot(df['age'])
```

```
<Axes: xlabel='age', ylabel='Density'>
```

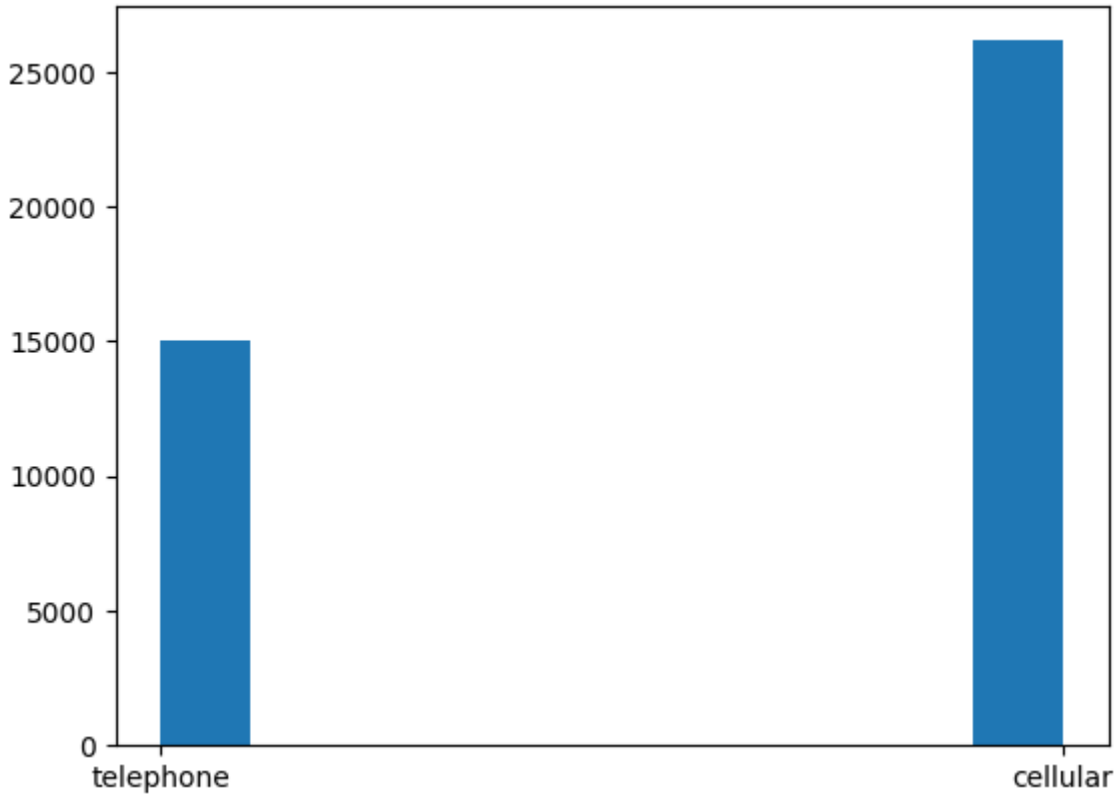



```
df['contact'].value_counts()
```

```
cellular     26144
telephone    15044
Name: contact, dtype: int64
```

```
plt.hist(df['contact'])
```

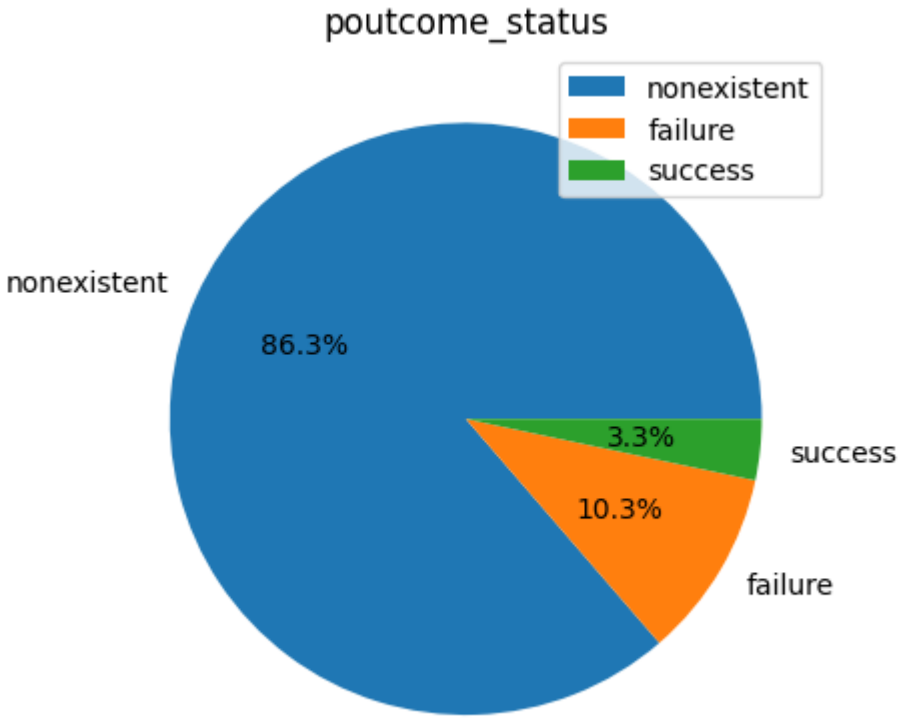
```
(array([15044.,     0.,     0.,     0.,     0.,     0.,     0.,     0.,
            0., 26144.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <BarContainer object of 10 artists>)
```



```
poutcome_counts=df['poutcome'].value_counts()
poutcome_counts
```

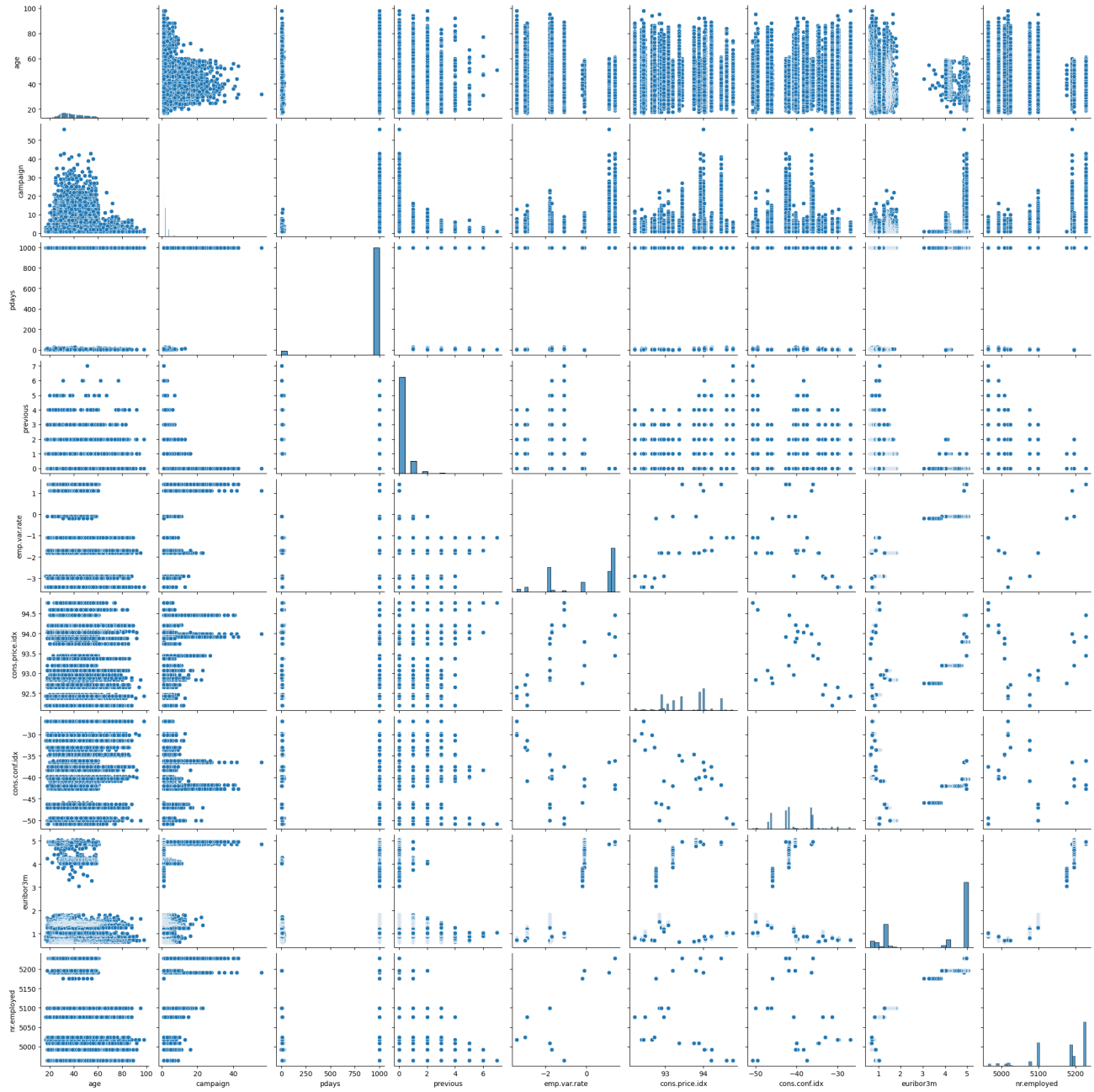
```
nonexistent    35563
failure         4252
success         1373
Name: poutcome, dtype: int64
```

```
mylabels=['nonexistent','failure','success']
plt.pie(poutcome_counts,labels=mylabels,autopct="%1.1f%%")
plt.title('poutcome_status')
plt.legend(loc='upper right')
plt.show()
```



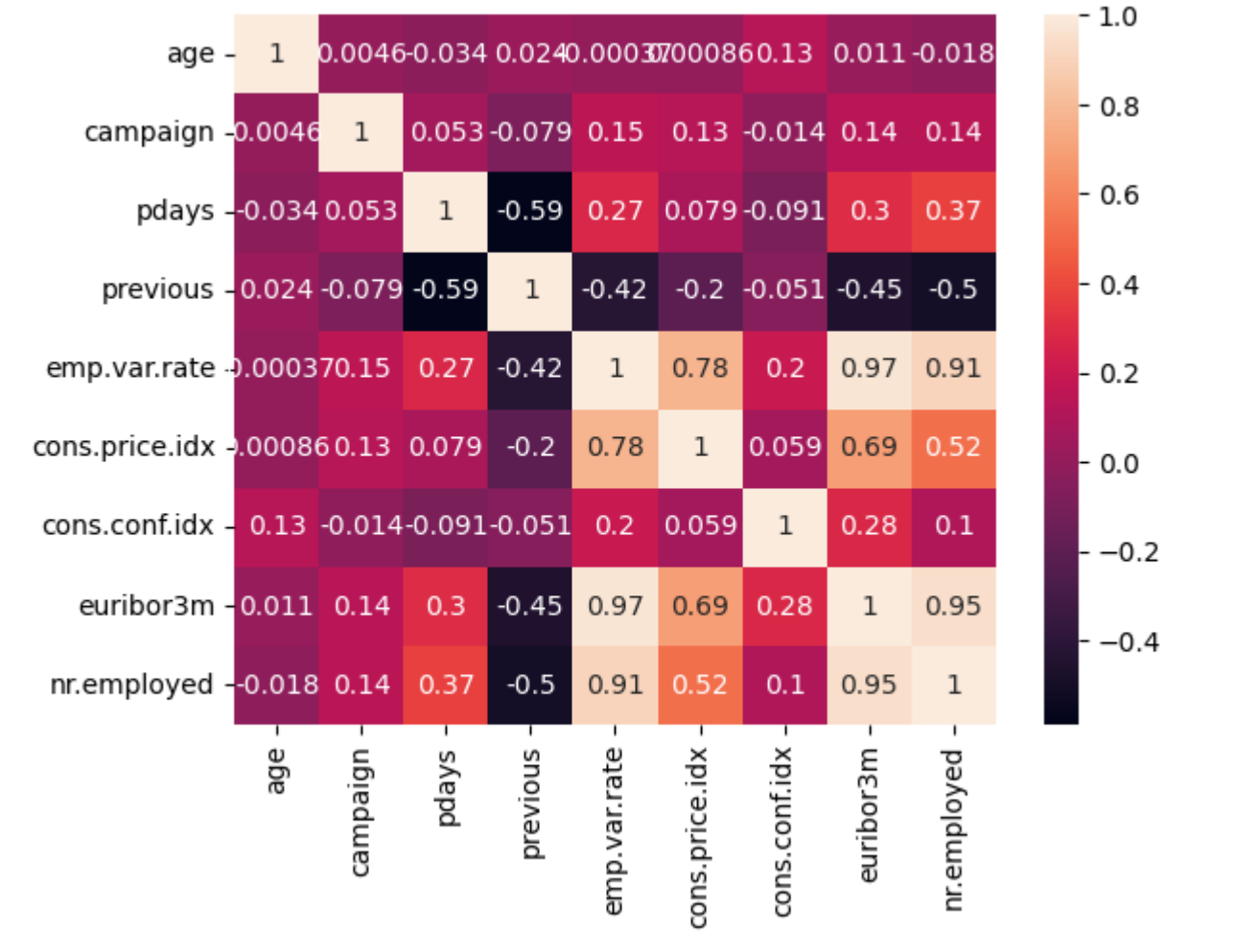

```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7a0bf52167d0>
```



```
sns.heatmap(df.corr(),annot=True)
```

```
<ipython-input-49-8df7bcac526d>:1: FutureWarning: The default value of numeric_only :
  sns.heatmap(df.corr(),annot=True)
<Axes: >
```



Converting strings to integer format using get dummies

```
df1=pd.get_dummies(df[['job','marital','education','default','housing','loan','contact',
df1
```

	job_blue-collar	job_entrepreneur	job_housemaid	job_management	job_retired	job_s empl
0	0	0	1	0	0	
1	0	0	0	0	0	
2	0	0	0	0	0	
3	0	0	0	0	0	
4	0	0	0	0	0	
...	...	...	...	...	...	
41183	0	0	0	0	1	
41184	1	0	0	0	0	
41185	0	0	0	0	1	
41186	0	0	0	0	0	
41187	0	0	0	0	1	

41188 rows × 43 columns

Combining df and df1

```
df2=pd.concat([df,df1],axis=1)
df2
```

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

41188 rows × 63 columns

```
df2=df2.drop(['job','marital','education','default','housing','loan','contact','month','
df2
```

	age	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	e
0	56	1	999	0	1.1	93.994	-36.4	
1	57	1	999	0	1.1	93.994	-36.4	
2	37	1	999	0	1.1	93.994	-36.4	
3	40	1	999	0	1.1	93.994	-36.4	
4	56	1	999	0	1.1	93.994	-36.4	
...	...	...	...	...	...	...	...	
41183	73	1	999	0	-1.1	94.767	-50.8	
41184	46	1	999	0	-1.1	94.767	-50.8	
41185	56	2	999	0	-1.1	94.767	-50.8	
41186	44	1	999	0	-1.1	94.767	-50.8	
41187	74	3	999	1	-1.1	94.767	-50.8	

41188 rows × 53 columns

```
df2.dtypes
```

```
age                             int64
campaign                        int64
pdays                           int64
previous                        int64
emp.var.rate                  float64
cons.price.idx                float64
cons.conf.idx                 float64
euribor3m                     float64
nr.employed                   float64
y                              object
job_blue-collar                 uint8
job_entrepreneur                uint8
job_housemaid                   uint8
job_management                  uint8
job_retired                     uint8
job_self-employed               uint8
job_services                    uint8
job_student                     uint8
job_technician                  uint8
job_unemployed                  uint8
job_unknown                     uint8
marital_married                 uint8
marital_single                  uint8
marital_unknown                 uint8
education_basic.6y              uint8
education_basic.9y              uint8
education_high.school           uint8
education_illiterate            uint8
```

```
education_professional.course      uint8
education_university.degree        uint8
education_unknown                  uint8
default_unknown                    uint8
default_yes                        uint8
housing_unknown                    uint8
housing_yes                        uint8
loan_unknown                       uint8
loan_yes                           uint8
contact_telephone                  uint8
month_aug                          uint8
month_dec                          uint8
month_jul                          uint8
month_jun                          uint8
month_mar                          uint8
month_may                          uint8
month_nov                          uint8
month_oct                          uint8
month_sep                          uint8
day_of_week_mon                    uint8
day_of_week_thu                    uint8
day_of_week_tue                    uint8
day_of_week_wed                    uint8
poutcome_nonexistent               uint8
poutcome_success                   uint8
dtype: object
```

Seperating x and y

```
x=df2.drop(['y'],axis=1)
y=df2['y']
```

Splitting training and testing data

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=42)
```

Performing Normalization

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(x_train)
x_train=scaler.fit_transform(x_train)
x_test=scaler.fit_transform(x_test)
```

Model Creation

```
from sklearn.tree import DecisionTreeClassifier
model=DecisionTreeClassifier(criterion='entropy')
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
y_pred
```

```
array(['no', 'no', 'no', ..., 'no', 'no', 'yes'], dtype=object)
```

```
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,Confus
print("accuracy score is",accuracy_score(y_test,y_pred))
```

```
accuracy score is 0.8320789835720644
```

```
print(classification_report(y_test,y_pred))
```

```
                  precision    recall  f1-score   support

              no       0.91      0.90      0.91     10968
             yes       0.27      0.29      0.28      1389

        accuracy                           0.83     12357
       macro avg       0.59      0.59      0.59     12357
    weighted avg       0.84      0.83      0.83     12357
```

```
labels=['no','yes']
result=confusion_matrix(y_test,y_pred)
cmd=ConfusionMatrixDisplay(result,display_labels=labels)
cmd.plot()
print(result)
```