PRODIGY TASK 3

In [1]:

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
import matplotlib.pyplot as plt
import seaborn as sns
get_ipython().run_line_magic('matplotlib', 'inline')
```

In [4]:

```
data = pd.read_csv("bank.csv")
```

In [5]:

```
data.head()
```

Out[5]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaig
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	

In [6]:

```
data.tail(10)
```

Out[6]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	ca
45201	53	management	married	tertiary	no	583	no	no	cellular	17	nov	226	
45202	34	admin.	single	secondary	no	557	no	no	cellular	17	nov	224	
45203	23	student	single	tertiary	no	113	no	no	cellular	17	nov	266	
45204	73	retired	married	secondary	no	2850	no	no	cellular	17	nov	300	
45205	25	technician	single	secondary	no	505	no	yes	cellular	17	nov	386	
45206	51	technician	married	tertiary	no	825	no	no	cellular	17	nov	977	
45207	71	retired	divorced	primary	no	1729	no	no	cellular	17	nov	456	
45208	72	retired	married	secondary	no	5715	no	no	cellular	17	nov	1127	
45209	57	blue-collar	married	secondary	no	668	no	no	telephone	17	nov	508	
45210	37	entrepreneur	married	secondary	no	2971	no	no	cellular	17	nov	361	

In [7]:

```
data.shape
```

Out[7]: (45211, 17)

In [8]:

```
print("Number of rows",data.shape[0])
print("Number of columns",data.shape[1])
```

```
Number of rows 45211
Number of columns 17
```

In [9]:

```
data.info()
```

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

In [15]:

```
print("let me know? ", data.isnull().values.any())
```

```
let me know?  False
```

In [17]:

```
data.isnull().sum()
```

Out[17]:

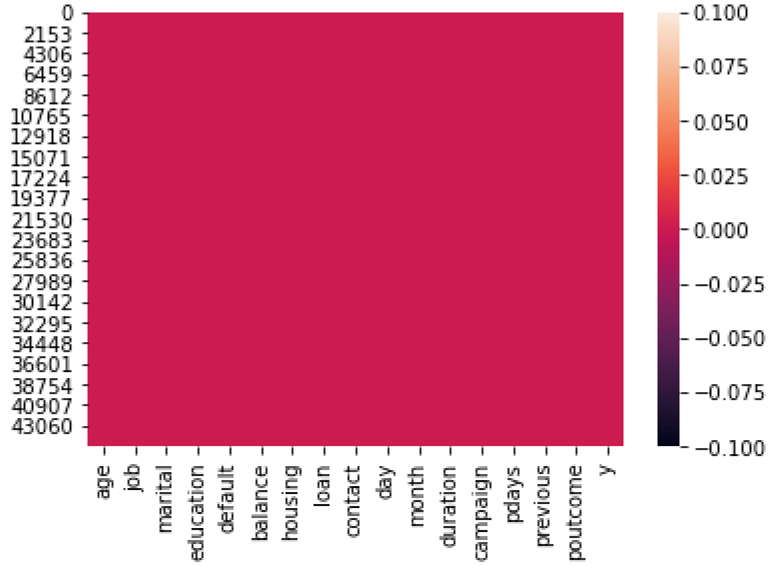
```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

In [19]:

```
sns.heatmap(data.isnull())
```

Out[19]:

```
<AxesSubplot:>
```



In [20]:
```
dup=data.duplicated().any()
```

In [21]:
```
print(dup)
```

```
False
```

In [23]:
```
data.describe()



data.describe(include='all')
```

Out[23]:

	age	job	marital	education	default	balance	housing	loan	contact	day
count	45211.000000	45211	45211	45211	45211	45211.000000	45211	45211	45211	45211.000000
unique	NaN	12	3	4	2	NaN	2	2	3	NaN
top	NaN	blue-collar	married	secondary	no	NaN	yes	no	cellular	NaN
freq	NaN	9732	27214	23202	44396	NaN	25130	37967	29285	NaN
mean	40.936210	NaN	NaN	NaN	NaN	1362.272058	NaN	NaN	NaN	15.806419
std	10.618762	NaN	NaN	NaN	NaN	3044.765829	NaN	NaN	NaN	8.322476
min	18.000000	NaN	NaN	NaN	NaN	-8019.000000	NaN	NaN	NaN	1.000000
25%	33.000000	NaN	NaN	NaN	NaN	72.000000	NaN	NaN	NaN	8.000000
50%	39.000000	NaN	NaN	NaN	NaN	448.000000	NaN	NaN	NaN	16.000000
75%	48.000000	NaN	NaN	NaN	NaN	1428.000000	NaN	NaN	NaN	21.000000
max	95.000000	NaN	NaN	NaN	NaN	102127.000000	NaN	NaN	NaN	31.000000

In [24]:
```
data['marital'] = data['marital'].map({'single': 2, 'married': 3, 'divorced': 3})
data['education'] = data['education'].map({'primary': 1, 'secondary': 2, 'tertiary': 3, 'u
data['default'] = data['default'].map({'no': 0, 'yes': 1})
data['housing'] = data['housing'].map({'no': 0, 'yes': 1})
data['loan'] = data['loan'].map({'no': 0, 'yes': 1})
data['y'] = data['y'].map({'no': 0, 'yes': 1})
```

In [26]:

```
data = data.drop(['job', 'contact', 'month', 'day', 'poutcome'], axis=1)
X = data.iloc[:, :-1]  # Features
y = data.iloc[:, -1]
```

In [29]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

In [30]:

```
clf = DecisionTreeClassifier()
```

In [31]:

```
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

In [32]:

```
accuracy = metrics.accuracy_score(y_test, y_pred)
precision = metrics.precision_score(y_test, y_pred)
recall = metrics.recall_score(y_test, y_pred)
f1_score = metrics.f1_score(y_test, y_pred)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1_score)
```

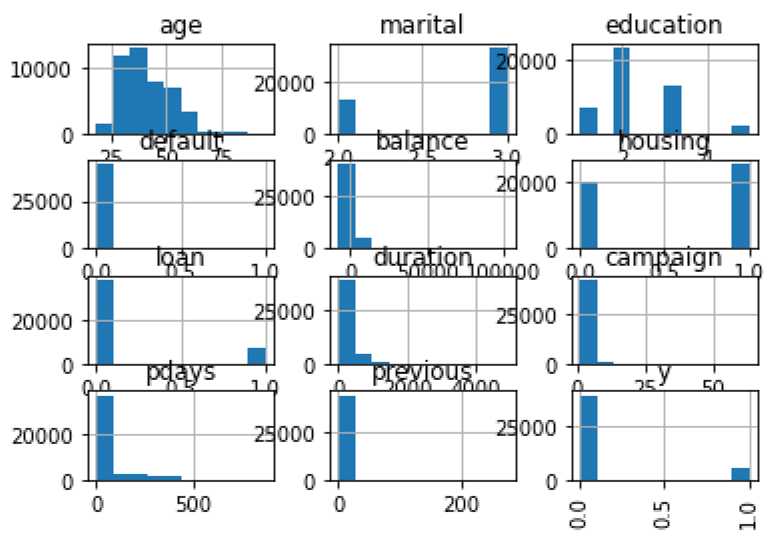
```
Accuracy: 0.8522559716897671
Precision: 0.3732512590934527
Recall: 0.4300451321727917
F1 Score: 0.3996405032953864
```

In [34]:

```
data.hist()
plt.xticks(rotation=90)
```

Out[34]:

```
(array([-0.5,  0. ,  0.5,  1. ,  1.5]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```

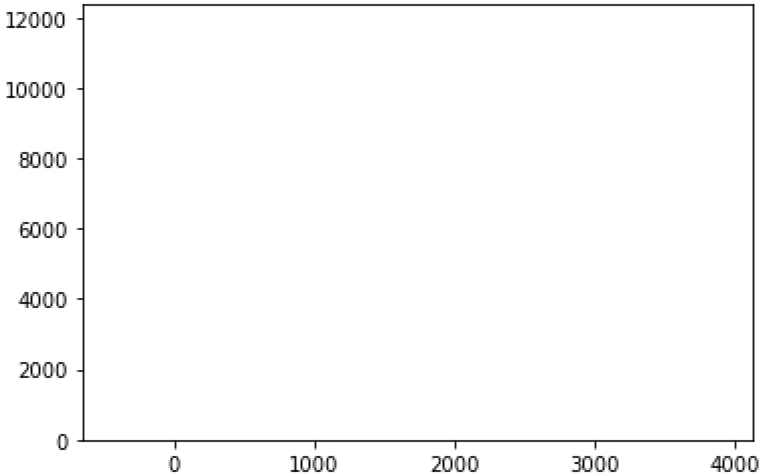


In [35]:

```
plt.hist(y_pred)

plt.hist(X_train.head(10))
```

Out[35]:
```
(array([[ 0., 10.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0., 10.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0., 10.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0., 10.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 2.,  3.,  3.,  0.,  0.,  1.,  0.,  0.,  0.,  1.],
        [ 0., 10.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0., 10.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0.,  9.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0., 10.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0., 10.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 0., 10.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.]]),
 array([-478. ,  -35.3,  407.4,  850.1, 1292.8, 1735.5, 2178.2, 2620.9,
        3063.6, 3506.3, 3949. ]),
 <a list of 11 BarContainer objects>)
```



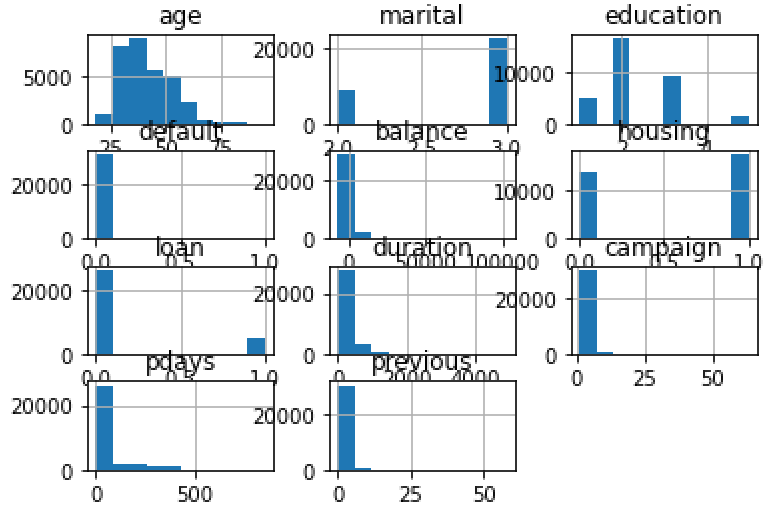
In [36]:
```
X_train.hist()

plt.hist(X_test.head(10))

plt.hist(X_test.tail(10))
```

Out[36]:
```
(array([[10.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [10.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [10.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [10.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 4.,  2.,  0.,  1.,  2.,  0.,  0.,  0.,  0.,  1.],
        [10.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [10.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 4.,  6.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [10.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [ 9.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
        [10.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.]]),
 array([-522.,  126.,  774., 1422., 2070., 2718., 3366., 4014., 4662.,
        5310., 5958.]),
 <a list of 11 BarContainer objects>)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js



In [37]:

```
sns.heatmap(X_train)

sns.heatmap(X_test)
```

Out[37]: <AxesSubplot:>