In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
df=pd.read_excel("credit.xlsx")
```

In [3]:

```
df
```

Out[3]:

	checking_balance	months_loan_duration	credit_history	purpose	amount	savings_balance	employment_duration	percent_of_income	years
0	< 0 DM	6.0	critical	furniture/appliances	1169.0	unknown	> 7 years	4.0	
1	1 - 200 DM	48.0	good	furniture/appliances	5951.0	< 100 DM	1 - 4 years	2.0	
2	unknown	12.0	critical	education	2096.0	< 100 DM	4 - 7 years	2.0	
3	< 0 DM	42.0	good	furniture/appliances	7882.0	< 100 DM	4 - 7 years	2.0	
4	< 0 DM	24.0	poor	car	4870.0	< 100 DM	1 - 4 years	3.0	
...	...	...	...	...	...	...	...	...	
995	unknown	12.0	good	furniture/appliances	1736.0	< 100 DM	4 - 7 years	3.0	
996	< 0 DM	30.0	good	car	3857.0	< 100 DM	1 - 4 years	4.0	
997	unknown	12.0	good	furniture/appliances	804.0	< 100 DM	> 7 years	4.0	
998	< 0 DM	45.0	good	furniture/appliances	1845.0	< 100 DM	1 - 4 years	4.0	
999	1 - 200 DM	45.0	critical	car	4576.0	100 - 500 DM	unemployed	3.0	

1000 rows × 17 columns

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   checking_balance      1000 non-null   object 
 1   months_loan_duration  1000 non-null   float64
 2   credit_history        1000 non-null   object 
 3   purpose               1000 non-null   object 
 4   amount                1000 non-null   float64
 5   savings_balance       1000 non-null   object 
 6   employment_duration   1000 non-null   object 
 7   percent_of_income     1000 non-null   float64
 8   years_at_residence    1000 non-null   float64
 9   age                   1000 non-null   float64
 10  other_credit          1000 non-null   object 
 11  housing               1000 non-null   object 
 12  existing_loans_count  1000 non-null   float64
 13  job                   1000 non-null   object 
 14  dependents            1000 non-null   float64
 15  phone                 1000 non-null   object 
 16  default               1000 non-null   object 
dtypes: float64(7), object(10)
memory usage: 132.9+ KB
```

In [5]:

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

Out[5]:

```
checking_balance        0
months_loan_duration    0
credit_history          0
purpose                 0
amount                  0
savings_balance         0
employment_duration     0
percent_of_income       0
years_at_residence      0
age                     0
other_credit            0
housing                 0
existing_loans_count    0
job                     0
dependents              0
phone                   0
default                 0
dtype: int64
```

In [6]:

```
cat_cols=df.select_dtypes(include="O").columns
cat_cols
```

Out[6]:

```
Index(['checking_balance', 'credit_history', 'purpose', 'savings_balance',
       'employment_duration', 'other_credit', 'housing', 'job', 'phone',
       'default'],
      dtype='object')
```

In [7]:

```
num_cols=df.select_dtypes(include=['int','float']).columns
num_cols
```

Out[7]:

```
Index(['months_loan_duration', 'amount', 'percent_of_income',
       'years_at_residence', 'age', 'existing_loans_count', 'dependents'],
      dtype='object')
```

In [8]:

```
for i in cat_cols:
    print("column name:",i)
    print("unique values:",df[i].unique())
    print("\n")
```

```
column name: checking_balance
unique values: ['< 0 DM' '1 - 200 DM' 'unknown' '> 200 DM']


column name: credit_history
unique values: ['critical' 'good' 'poor' 'perfect' 'very good']


column name: purpose
unique values: ['furniture/appliances' 'education' 'car' 'business' 'renovations' 'car0']


column name: savings_balance
unique values: ['unknown' '< 100 DM' '500 - 1000 DM' '> 1000 DM' '100 - 500 DM']


column name: employment_duration
unique values: ['> 7 years' '1 - 4 years' '4 - 7 years' 'unemployed' '< 1 year']


column name: other_credit
unique values: ['none' 'bank' 'store']


column name: housing
unique values: ['own' 'other' 'rent']


column name: job
unique values: ['skilled' 'unskilled' 'management' 'unemployed']


column name: phone
unique values: ['yes' 'no']


column name: default
unique values: ['no' 'yes']
```

In [9]:

```
df['purpose']=df['purpose'].replace("car0","car")
```

In [10]:

```
for i in num_cols:
    print("column name:",i)
    print("unique values:",df[i].unique())
```

```
column name: amount
unique values: [ 1169.  5951.  2096.  7882.  4870.  9055.  2835.  6948.  3059.  5234.
  1295.  4308.  1567.  1199.  1403.  1282.  2424.  8072. 12579.  3430.
  2134.  2647.  2241.  1804.  2069.  1374.   426.   409.  2415.  6836.
  1913.  4020.  5866.  1264.  1474.  4746.  6110.  2100.  1225.   458.
  2333.  1158.  6204.  6187.  6143.  1393.  2299.  1352.  7228.  2073.
  5965.  1262.  3378.  2225.   783.  6468.  9566.  1961.  6229.  1391.
  1537.  1953. 14421.  3181.  5190.  2171.  1007.  1819.  2394.  8133.
   730.  1164.  5954.  1977.  1526.  3965.  4771.  9436.  3832.  5943.
  1213.  1568.  1755.  2315.  1412. 12612.  2249.  1108.   618.  1409.
   797.  3617.  1318. 15945.  2012.  2622.  2337.  7057.  1469.  2323.
   932.  1919.  2445. 11938.  6458.  6078.  7721.  1410.  1449.   392.
  6260.  7855.  1680.  3578.  7174.  2132.  4281.  2366.  1835.  3868.
  1768.   781.  1924.  2121.   701.   639.  1860.  3499.  8487.  6887.
  2708.  1984. 10144.  1240.  8613.   766.  2728.  1881.   709.  4795.
  3416.  2462.  2288.  3566.   860.   682.  5371.  1582.  1346.  5848.
  7758.  6967.  1288.   339.  3512.  1898.  2872.  1055.  7308.   909.
  2978.  1131.  1577.  3972.  1935.   950.   763.  2064.  1414.  3414.
  7485.  2577.   338.  1963.   571.  9572.  4455.  1647.  3777.   884.
  1360.  5129.  1175.   674.  3244.  4591.  3844.  3915.  2108.  3031.
```

In [11]:

```
for i in num_cols:
    df[i]=df[i].astype("int")
```

In [12]:

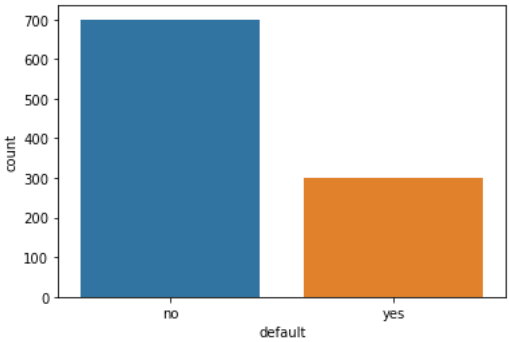
```
df.describe()
```

Out[12]:

	months_loan_duration	amount	percent_of_income	years_at_residence	age	existing_loans_count	dependents
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000
mean	20.903000	3271.258000	2.973000	2.845000	35.546000	1.407000	1.155000
std	12.058814	2822.736876	1.118715	1.103718	11.375469	0.577654	0.362086
min	4.000000	250.000000	1.000000	1.000000	19.000000	1.000000	1.000000
25%	12.000000	1365.500000	2.000000	2.000000	27.000000	1.000000	1.000000
50%	18.000000	2319.500000	3.000000	3.000000	33.000000	1.000000	1.000000
75%	24.000000	3972.250000	4.000000	4.000000	42.000000	2.000000	1.000000
max	72.000000	18424.000000	4.000000	4.000000	75.000000	4.000000	2.000000

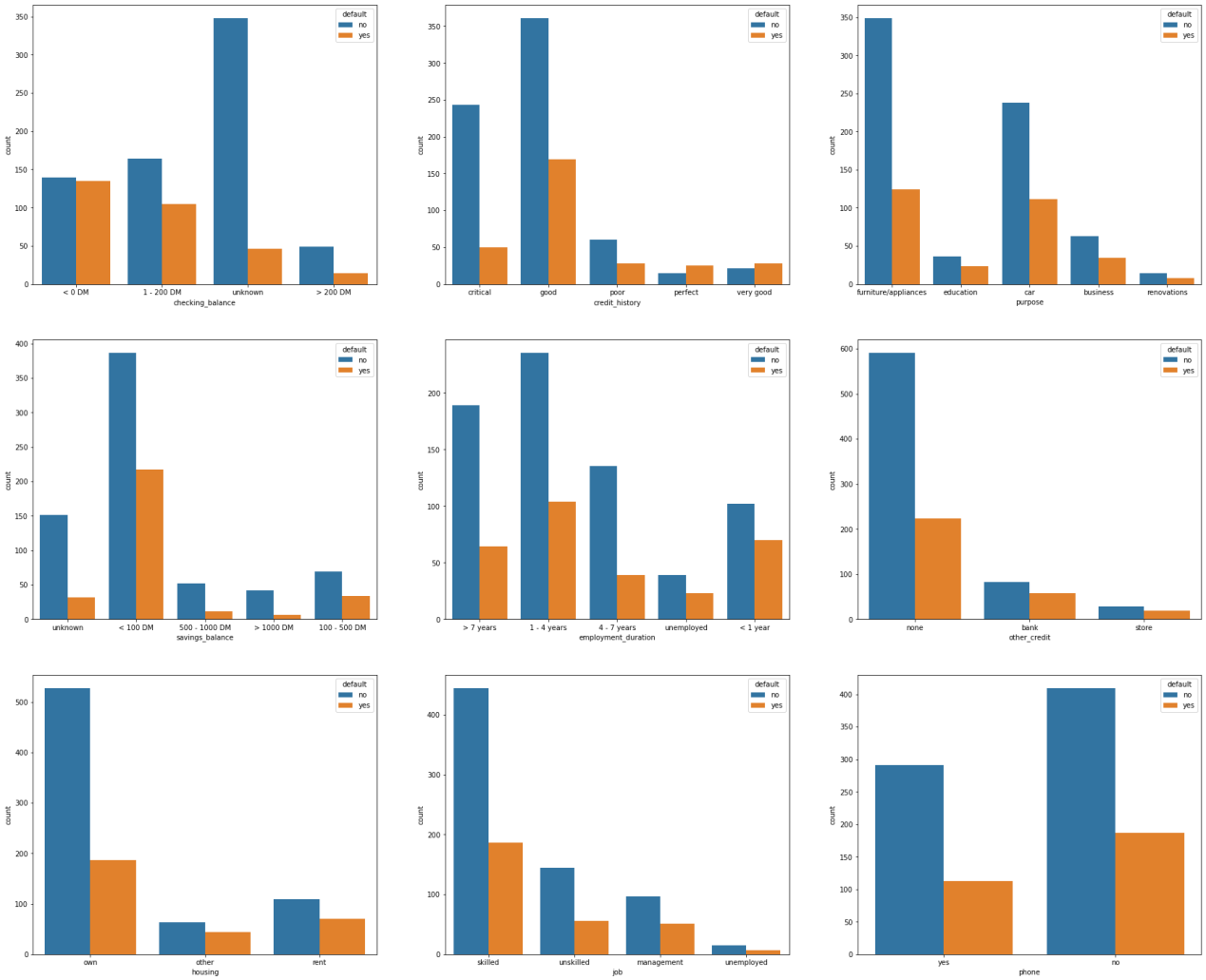
In [13]:

```
sns.countplot(df['default'])
plt.show()
```



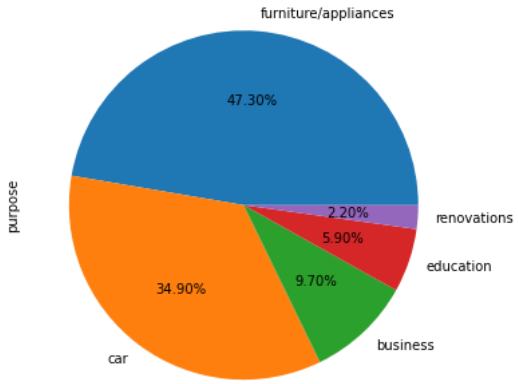
In [14]:

```
plt.figure(figsize=(30,25))
count=1
for i in cat_cols:
    if i!='default':
        plt.subplot(3,3,count)
        sns.countplot(df[i],hue=df['default'])
        count+=1
```



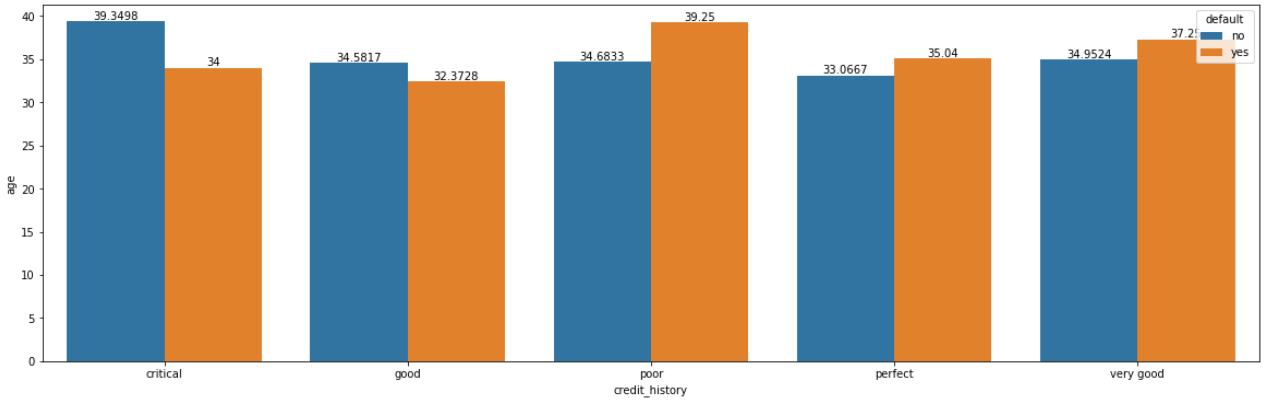
In [15]:

```
plt.figure(figsize=(10,6))
df['purpose'].value_counts().plot.pie(autopct="%.2f%%")
plt.show()
```



In [16]:

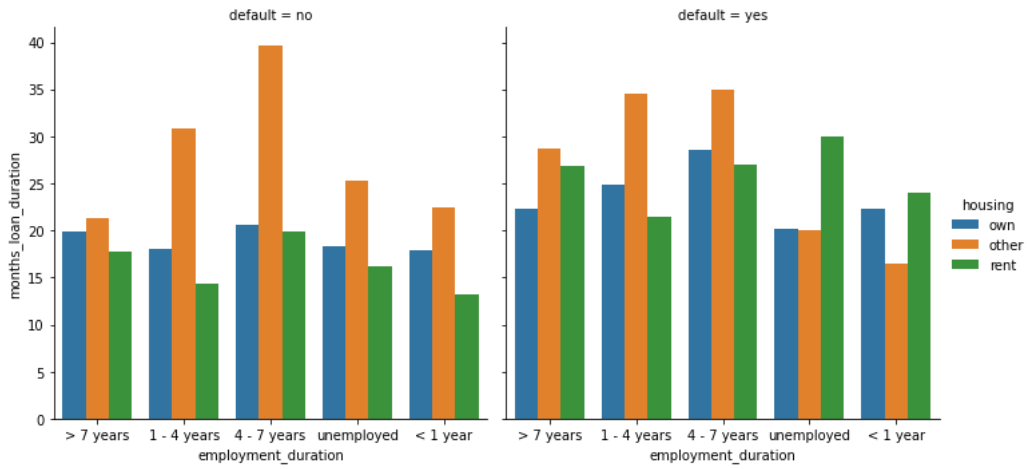
```
plt.figure(figsize=(20,6))
ax=sns.barplot(df['credit_history'],df['age'],hue=df['default'],ci=False)
ax.bar_label(ax.containers[0])
ax.bar_label(ax.containers[1])
plt.show()
```



In [17]:

```
plt.figure(figsize=(20,6))
sns.catplot(x='employment_duration',y='months_loan_duration',hue='housing',col='default',kind='bar',data=df,ci=False)
plt.show()
```

```
<Figure size 1440x432 with 0 Axes>
```



In [18]:

```
df['employment_duration'].unique()
```

Out[18]:

```
array(['> 7 years', '1 - 4 years', '4 - 7 years', 'unemployed',
       '< 1 year'], dtype=object)
```

In [19]:

```
df.replace({'employment_duration':{'unemployed':"0",'< 1 year':"1","1 - 4 years":"2","4 - 7 years":"3","> 7 years":"4"}},inplace=True)
```

In [20]:

```
df['employment_duration']=df['employment_duration'].astype("int")
```

In [21]:

```
cat_cols=df.select_dtypes(include="O").columns
cat_cols
```

Out[21]:

```
Index(['checking_balance', 'credit_history', 'purpose', 'savings_balance',
       'other_credit', 'housing', 'job', 'phone', 'default'],
      dtype='object')
```

In [22]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for i in cat_cols:
    df[i]=le.fit_transform(df[i])
```

In [23]:

```
x=df.drop('default',axis=1)
```

In [24]:

```
y=df['default']
```

In [25]:

```
from sklearn.model_selection import train_test_split
```

In [26]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=123)
```

In [27]:

```
x_train.shape
```

Out[27]:

```
(800, 16)
```

In [28]:

```
x_test.shape
```

Out[28]:

```
(200, 16)
```

In [29]:

```
y_train.shape
```

Out[29]:

```
(800,)
```

In [30]:

```
y_test.shape
```

Out[30]:

```
(200,)
```

In [31]:

```
from sklearn.preprocessing import StandardScaler
```

In [32]:

```
sc=StandardScaler()
```

In [33]:

```
x_train=sc.fit_transform(x_train)
```

In [34]:

```
x_test=sc.transform(x_test)
```

In [35]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [36]:

```
dt=DecisionTreeClassifier()
```

In [37]:

```
from sklearn.metrics import precision_score
```

In [38]:

```
def my_model(model):
    model.fit(x_train,y_train)
    y_pred_train=model.predict(x_train)
    y_pred_test=model.predict(x_test)
    print("Train Data")
    print(round(precision_score(y_train,y_pred_train),2))
    print("Test Data")
    print(round(precision_score(y_test,y_pred_test),2))
```

In [39]:

```
my_model(dt)
```

```
Train Data
1.0
Test Data
0.6
```

In [45]:

```
for i in range(1,15):
    dt=DecisionTreeClassifier(max_depth=i)
    dt.fit(x_train,y_train)
    y_pred_train=dt.predict(x_train)
    y_pred_test=dt.predict(x_test)
    print('When max_depth is:',i)
    print("Train Data")
    print(round(precision_score(y_train,y_pred_train),2))
    print("Test Data")
    print(round(precision_score(y_test,y_pred_test),2))
    print('*'*90)
```

```
When max_depth is: 1
Train Data
0.0
Test Data
0.0
**********************************************************************************************
When max_depth is: 2
Train Data
0.56
Test Data
0.57
**********************************************************************************************
When max_depth is: 3
Train Data
0.64
Test Data
0.64
**********************************************************************************************
When max_depth is: 4
Train Data
0.64
Test Data
0.63
**********************************************************************************************
When max_depth is: 5
Train Data
0.66
Test Data
0.63
**********************************************************************************************
When max_depth is: 6
Train Data
0.77
Test Data
0.59
**********************************************************************************************
When max_depth is: 7
Train Data
0.81
Test Data
0.61
**********************************************************************************************
When max_depth is: 8
Train Data
0.83
Test Data
0.59
**********************************************************************************************
When max_depth is: 9
Train Data
0.89
Test Data
0.61
**********************************************************************************************
When max_depth is: 10
Train Data
0.93
Test Data
0.63
**********************************************************************************************
When max_depth is: 11
Train Data
0.94
Test Data
0.59
**********************************************************************************************
When max_depth is: 12
Train Data
0.96
Test Data
0.6
**********************************************************************************************
When max_depth is: 13
Train Data
0.97
Test Data
0.57
**********************************************************************************************
When max_depth is: 14
Train Data
0.99
Test Data
0.57
**********************************************************************************************
```

In [40]:

```
param_grid={
    'criterion':['gini','entropy'],
    'max_depth':np.arange(3,51),
    'min_samples_split':np.arange(1,51),
    'min_samples_leaf':np.arange(1,51),
    'max_features':['log','auto',None]
}
```

In [41]:

```
from sklearn.model_selection import GridSearchCV
grid_cv=GridSearchCV(dt,param_grid=param_grid,cv=5,scoring="precision",n_jobs=-1)
```

In [42]:

```
my_model(grid_cv)
```

```
Train Data
0.73
Test Data
0.56
```