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

In [5]:
```
df = pd.read_csv("Churn_Modelling[1].csv")
```

In [6]:
```
df.isnull()
```

Out[6]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
9995	False	False	False	False	False	False	False	False
9996	False	False	False	False	False	False	False	False
9997	False	False	False	False	False	False	False	False
9998	False	False	False	False	False	False	False	False
9999	False	False	False	False	False	False	False	False

10000 rows × 14 columns

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   RowNumber        10000 non-null  int64  
 1   CustomerId       10000 non-null  int64  
 2   Surname          10000 non-null  object 
 3   CreditScore      10000 non-null  int64  
 4   Geography        10000 non-null  object 
 5   Gender           10000 non-null  object 
 6   Age              10000 non-null  int64  
 7   Tenure           10000 non-null  int64  
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64  
 10  HasCrCard        10000 non-null  int64  
 11  IsActiveMember   10000 non-null  int64  
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64  
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```

In [10]:
```
df.dtypes
```

Out[10]:
```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
```

In [11]:
```
df.columns
```

Out[11]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

In [14]:
```
df=df.drop(['RowNumber', 'CustomerId', 'Surname'], axis =1)
```

In [15]:
```
df.head()
```

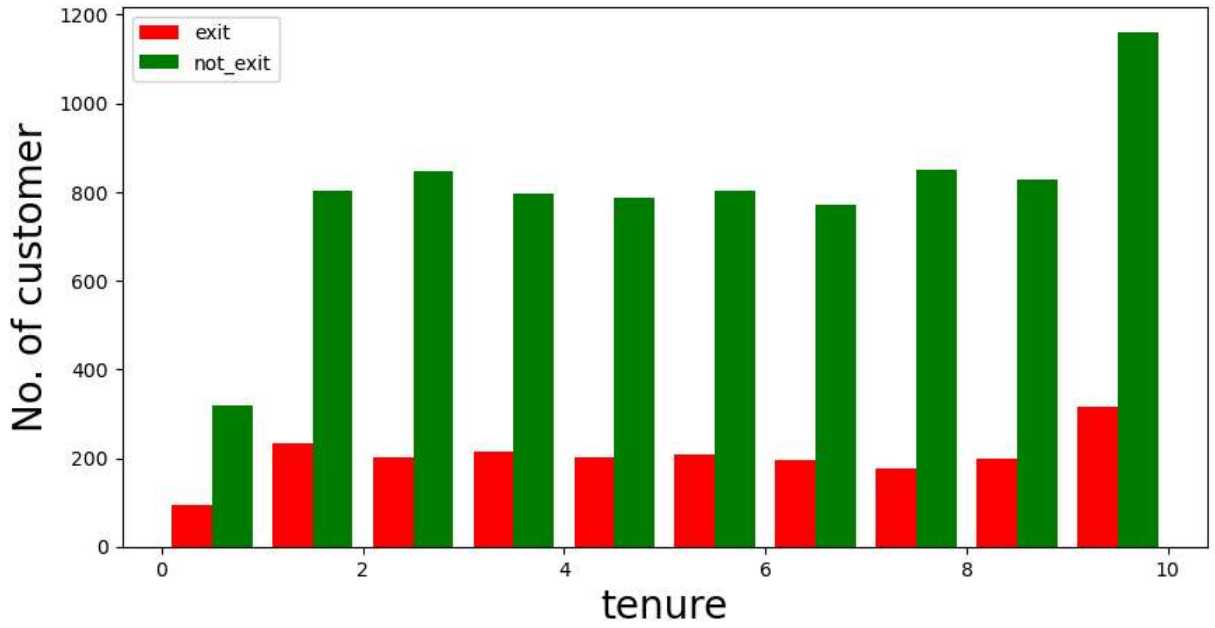
Out[15]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard
0	619	France	Female	42	2	0.00	1	1
1	608	Spain	Female	41	1	83807.86	1	0
2	502	France	Female	42	8	159660.80	3	1
3	699	France	Female	39	1	0.00	2	0
4	850	Spain	Female	43	2	125510.82	1	1

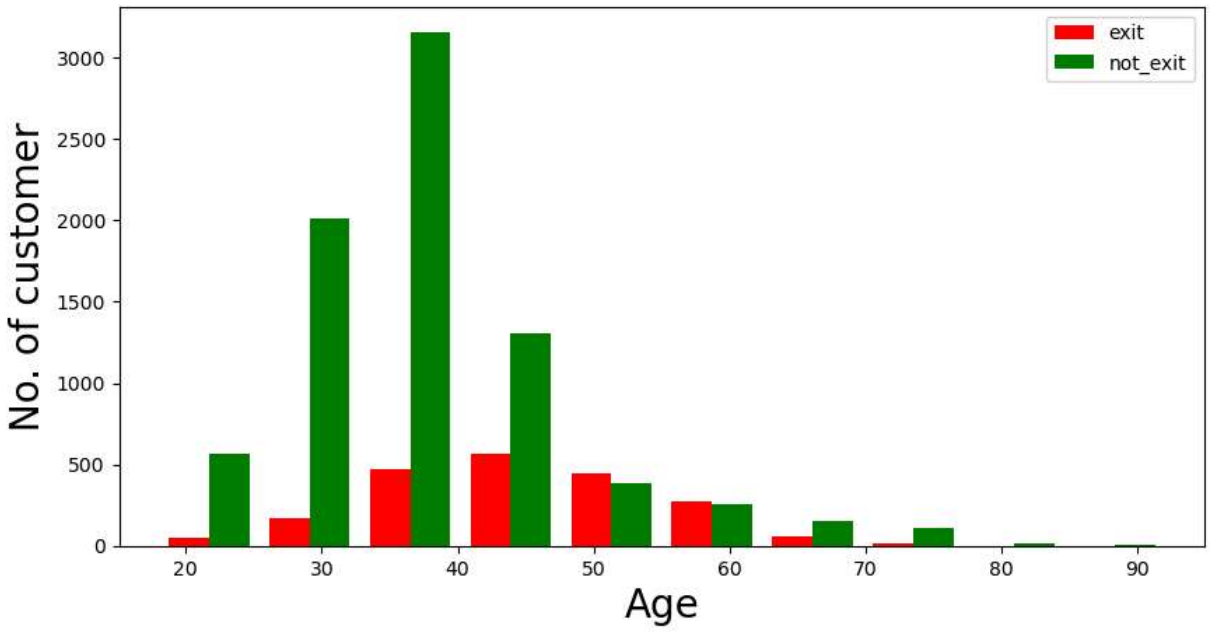
In [21]:
```
def visualization (x, y, xlabel):
    plt.figure(figsize=(10,5))
    plt.hist([x, y], color=['red', 'green'], label = ['exit','not_exit'])
    plt.xlabel(xlabel,fontsize=20)
    plt.ylabel("No. of customer",fontsize=20)
    plt.legend()
```

In [22]:
```
df_churn_exited = df[df['Exited']==1]['Tenure']
df_churn_not_exited = df[df['Exited']==0]['Tenure']
```

In [23]:
```
visualization(df_churn_exited,df_churn_not_exited, "tenure")
```



In [24]:
```
df_churn_exited2 = df[df['Exited']==1]['Age']
df_churn_not_exited2 = df[df['Exited']==0]['Age']
visualization(df_churn_exited2,df_churn_not_exited2, "Age")
```



In [25]:
```
x=df[['CreditScore','Gender','Age','Tenure','Balance','NumOfProducts','HasCrCard','
states = pd.get_dummies(df['Geography'],drop_first = True)
gender = pd.get_dummies(df['Gender'],drop_first = True)
```

In [26]:
```
df.head()
```

Out[26]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard
0	619	France	Female	42	2	0.00	1	1
1	608	Spain	Female	41	1	83807.86	1	0
2	502	France	Female	42	8	159660.80	3	1
3	699	France	Female	39	1	0.00	2	0
4	850	Spain	Female	43	2	125510.82	1	1

In [29]:
```
x= df[['CreditScore','Age','Tenure','Balance','NumOfProducts','HasCrCard','IsActive
y=df['Exited']
```

In [31]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.30)
```

from sklearn.preprocessing import StandardScaler sc =StandardScaler() x_train = sc.fit_transform(x_train) x_test = sc.transform(x_test)

In [32]:
```
x_train
```

Out[32]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember
1870	624	33	6	0.00	2	0	0
6735	842	37	4	132446.08	2	1	0
4485	751	34	9	108513.25	2	1	1
5459	530	36	7	0.00	2	1	0
9860	775	30	10	191091.74	2	1	1
...	...	...	...	...	...	...	...
9527	850	40	9	99816.46	1	1	1
2099	548	57	6	76165.65	1	1	1
8228	554	39	10	160132.75	1	1	0
7327	650	42	4	194532.66	1	1	0
5360	633	35	10	0.00	2	1	0

7000 rows × 8 columns

In [33]:
```
x_test
```

Out[33]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember
160	717	22	6	101060.25	1	0	1
7211	639	37	4	116121.84	2	0	1
4800	690	39	6	0.00	2	1	0
1338	625	52	5	164978.01	1	1	1
902	645	48	7	90612.34	1	1	1
...	...	...	...	...	...	...	...
9828	576	39	1	0.00	2	1	1
4479	598	47	2	0.00	2	1	1
4109	702	28	1	103033.83	1	1	1
1250	548	32	5	175214.71	1	1	1
8714	703	41	6	109941.51	1	1	0

3000 rows × 8 columns

In [35]:
```
from sklearn.neural_network import MLPClassifier
ann = MLPClassifier(hidden_layer_sizes=(100,100,100),random_state =0,max_iter=100,a
```

```
ann.fit(x_train,y_train)
```

Out[35]:

MLPClassifier

```
MLPClassifier(hidden_layer_sizes=(100, 100, 100), max_iter=100, random_sta
te=0)
```

In [36]:
```
y_pred=ann.predict(x_test)
```

In [37]:
```
y_pred
```

Out[37]:
```
array([1, 1, 1, ..., 0, 1, 1], dtype=int64)
```

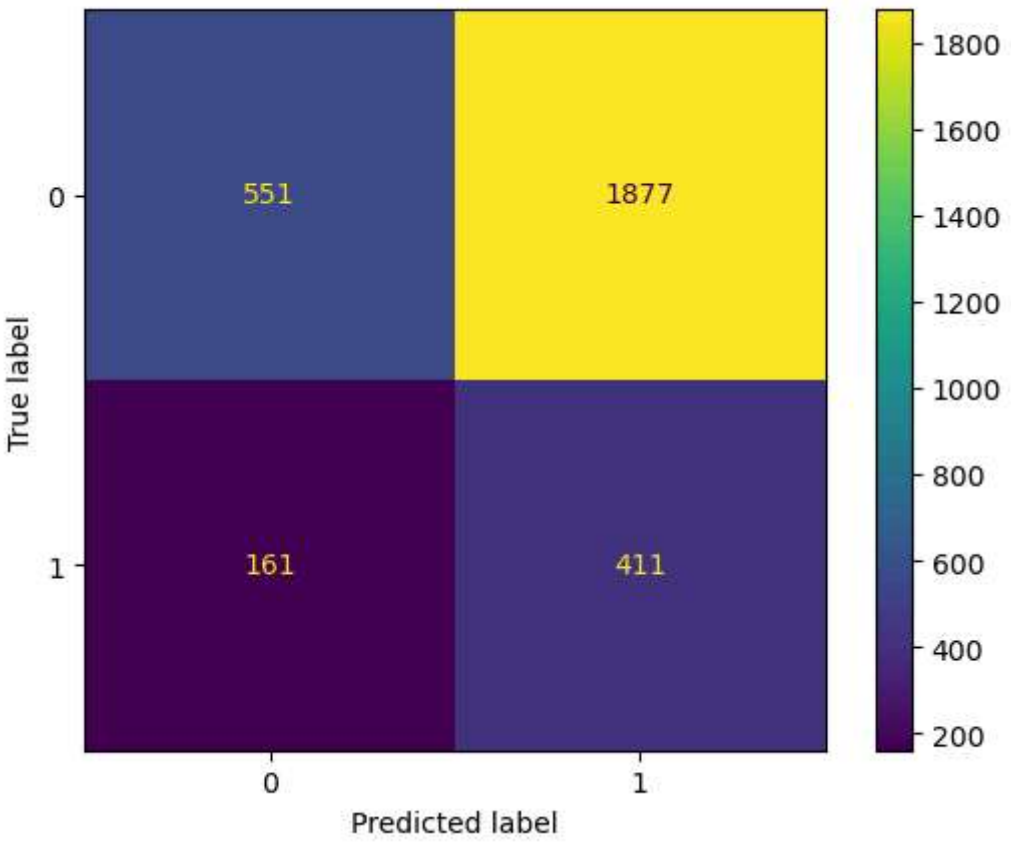
In [39]:
```
from sklearn.metrics import ConfusionMatrixDisplay,accuracy_score,classification_re
```

In [40]:
```
y_test.value_counts()
```

Out[40]:
```
Exited
0    2428
1     572
Name: count, dtype: int64
```

In [41]:
```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred)
```

Out[41]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2587d433010>
```



In [42]:
```
print(accuracy_score(y_test,y_pred))
```

```
0.32066666666666666
```

In [44]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.77      0.23      0.35      2428
           1       0.18      0.72      0.29       572

    accuracy                           0.32      3000
   macro avg       0.48      0.47      0.32      3000
weighted avg       0.66      0.32      0.34      3000
```

In [ ]:
```

```