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

from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score, classification_report

import warnings
warnings.filterwarnings("ignore")
import plotly.express as px
import plotly.io as pio


df=pd.read_csv("/content/bank (1).csv")


df
```

	age	job	marital	education	default	balance	housing	loan	cont
0	59	0	1	1	0	2343	1	0	
1	56	0	1	1	0	45	0	0	
2	41	9	1	1	0	1270	1	0	
3	55	7	1	1	0	2476	1	0	
4	54	0	1	2	0	184	0	0	
...	...	...	...	...	...	...	...	...	
11157	33	1	2	0	0	1	1	0	
11158	39	7	1	1	0	733	0	0	
11159	32	9	2	1	0	29	0	0	
11160	43	9	1	1	0	0	0	1	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11162 entries, 0 to 11161
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        11162 non-null  int64
 1   job        11162 non-null  int64
 2   marital    11162 non-null  int64
 3   education  11162 non-null  int64
 4   default    11162 non-null  int64
 5   balance    11162 non-null  int64
 6   housing    11162 non-null  int64
 7   loan       11162 non-null  int64
 8   contact    11162 non-null  int64
 9   day        11162 non-null  int64
 10  month      11162 non-null  int64
 11  duration   11162 non-null  int64
 12  campaign   11162 non-null  int64
 13  pdays      11162 non-null  int64
 14  previous   11162 non-null  int64
 15  poutcome   11162 non-null  int64
 16  deposit    11162 non-null  int64
dtypes: int64(17)
memory usage: 1.4 MB
```

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

```
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
deposit      0
dtype: int64
```

```
df.describe()
```

	age	job	marital	education	default
count	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000
mean	41.231948	4.487905	1.199337	1.285164	0.015051
std	11.913369	3.225132	0.625552	0.749478	0.121761
min	18.000000	0.000000	0.000000	0.000000	0.000000
25%	32.000000	1.000000	1.000000	1.000000	0.000000
50%	39.000000	4.000000	1.000000	1.000000	0.000000
75%	49.000000	7.000000	2.000000	2.000000	0.000000
max	95.000000	11.000000	2.000000	3.000000	1.000000

```
df.corr()
```

	age	job	marital	education	default	balance	
age	1.000000	-0.031603	-0.442782	-0.126018	-0.011425	0.112300	-(
job	-0.031603	1.000000	0.078314	0.147046	-0.007066	0.028736	-(
marital	-0.442782	0.078314	1.000000	0.125845	-0.014691	-0.002138	-(
education	-0.126018	0.147046	0.125845	1.000000	-0.010709	0.051728	-(
default	-0.011425	-0.007066	-0.014691	-0.010709	1.000000	-0.060954	(
balance	0.112300	0.028736	-0.002138	0.051728	-0.060954	1.000000	-(
housing	-0.168700	-0.136965	-0.036345	-0.109168	0.011076	-0.077092	1
loan	-0.031418	-0.067092	-0.062029	-0.073154	0.076434	-0.084589	(
contact	0.027762	-0.087915	-0.060456	-0.132540	0.035709	-0.027295	(
day	-0.000762	0.026589	-0.003642	0.016759	0.017342	0.010467	-(
month	-0.026130	-0.076011	-0.004070	-0.055868	0.000950	0.007264	(
duration	0.000189	0.002432	0.006781	-0.019122	-0.009760	0.022436	(
campaign	-0.005278	0.003104	-0.030794	-0.005327	0.030975	-0.013894	(
pdays	0.002774	-0.003385	0.031200	0.025165	-0.036282	0.017411	(
previous	0.020169	0.012665	0.031281	0.022427	-0.035273	0.030805	-(

```
df.drop(["job","marital","education","default","contact","campaign"],axis=1,inplace=True)
```

```
df.head()
```

	age	balance	housing	loan	day	month	duration	pdays	previous	pou
0	59	2343	1	0	5	8	1042	-1	0	
1	56	45	0	0	5	8	1467	-1	0	
2	41	1270	1	0	5	8	1389	-1	0	
3	55	2476	1	0	5	8	579	-1	0	

```
df.keys
```

```
<bound method NDFrame.keys of        age  balance  housing  loan  day  month  duration  pdays  previous  \
0       59     2343        1     0    5      8      1042     -1         0
1       56       45        0     0    5      8      1467     -1         0
2       41     1270        1     0    5      8      1389     -1         0
3       55     2476        1     0    5      8       579     -1         0
4       54      184        0     0    5      8       673     -1         0
...    ...      ...      ...   ...  ...    ...       ...    ...       ...
11157   33        1        1     0   20      0       257     -1         0
11158   39      733        0     0   16      6        83     -1         0
11159   32       29        0     0   19      1       156     -1         0
11160   43        0        0     1    8      8         9    172         5
11161   34        0        0     0    9      5       628     -1         0

       poutcome  deposit
0             3        1
1             3        1
2             3        1
3             3        1
4             3        1
...         ...      ...
11157         3        0
11158         3        0
11159         3        0
11160         0        0
11161         3        0

[11162 rows x 11 columns]>
```

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

```
<seaborn.axisgrid.PairGrid at 0x7f55e2e553a0>
```

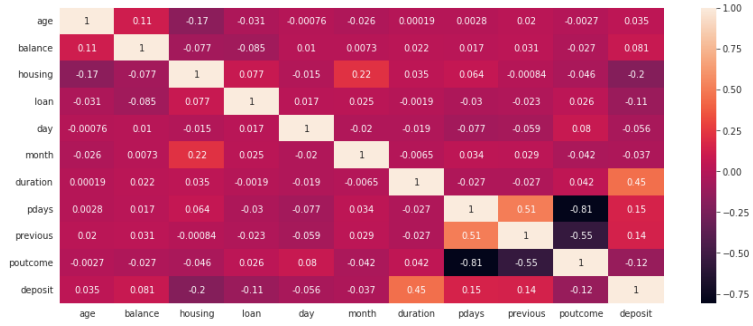
```
df.isin(df)
```

	age	balance	housing	loan	day	month	duration	pdays	previous

```
df.interpolate()
```

	age	balance	housing	loan	day	month	duration	pdays	previous
0	59	2343	1	0	5	8	1042	-1	0
1	56	45	0	0	5	8	1467	-1	0
2	41	1270	1	0	5	8	1389	-1	0
3	55	2476	1	0	5	8	579	-1	0
4	54	184	0	0	5	8	673	-1	0
...	...	...	...	...	...	...	...	...	...
11157	33	1	1	0	20	0	257	-1	0
11158	39	733	0	0	16	6	83	-1	0
11159	32	29	0	0	19	1	156	-1	0
11160	43	0	0	1	8	8	9	172	5

```
plt.figure(figsize= (15,6))
sns.set_style("darkgrid")
sns.heatmap(df.corr(),annot= True)
plt.show()
```

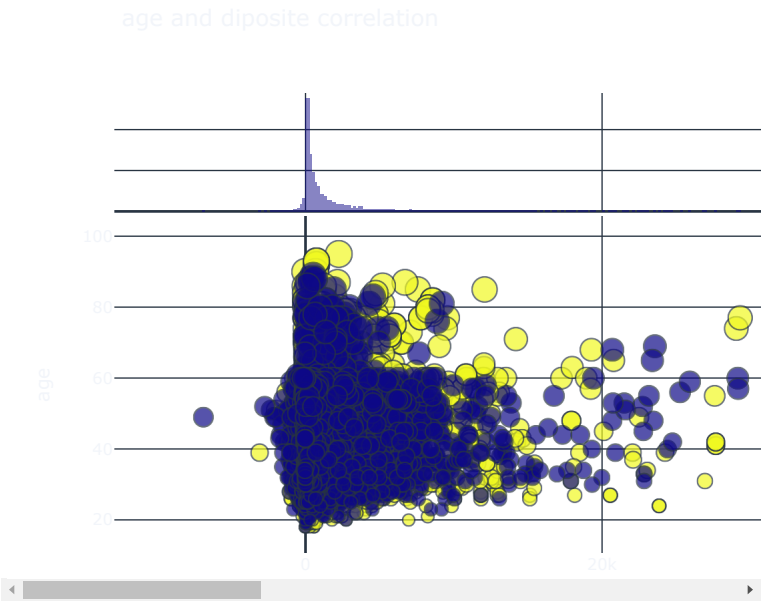



```
ax= px.histogram(df,x= "loan", template= "plotly_dark",color= "previous",title='Age distribution')
ax.show()
```

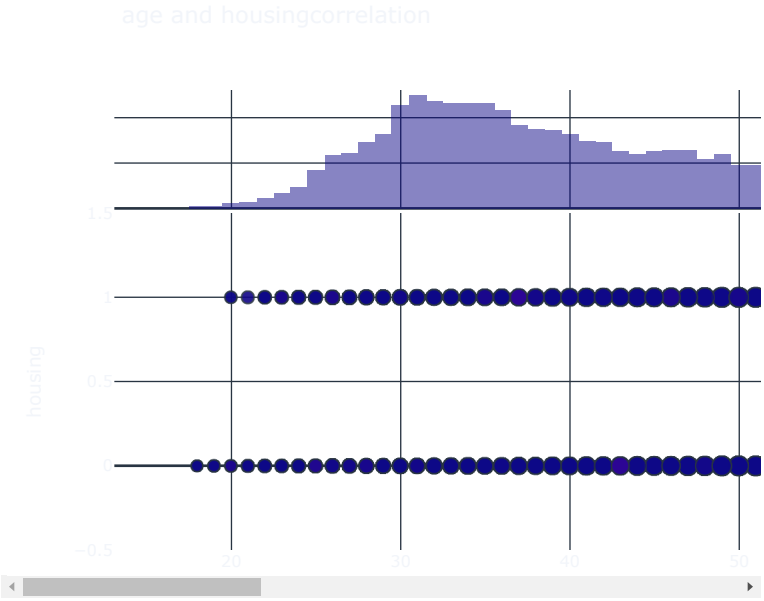

```
ax= px.pie(df, names= "housing",template= "plotly_dark",title= "housing loan distrbusion",hole= 0.5)
ax.show()
```



```
ax= px.scatter(df,x= "balance",y= "age",marginal_x='histogram', marginal_y='histogram',size="age", size_max=20,
               template= "plotly_dark",color= "deposit",title="age and diposite correlation")
ax.show()
```

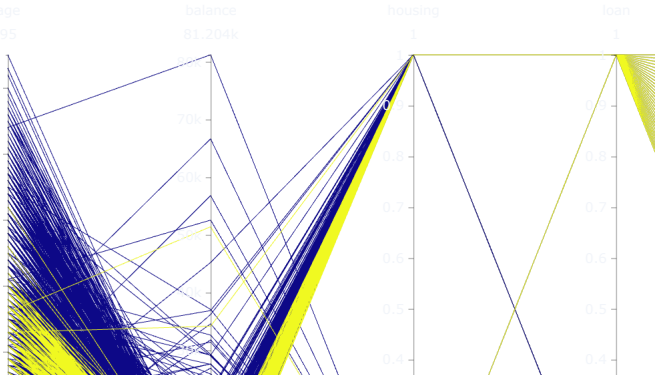



```
ax= px.scatter(df,x= "age",y= "housing",marginal_x='histogram', marginal_y='histogram',size="age", size_max=20,
               template= "plotly_dark",color= "previous",title="age and housingcorrelation")
ax.show()
```

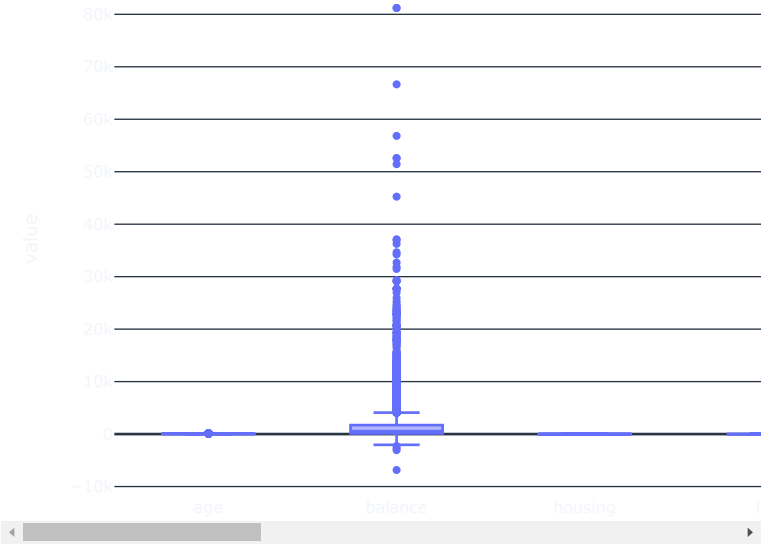



```
ax = px.scatter_3d(df, x="age", y="loan", z="deposit",template= "plotly_dark",color="month")
ax.show()
```

```
ax= px.parallel_coordinates(df, color="loan",template= "plotly_dark")
ax.show()
```




```
ax = px.box(df,template= "plotly_dark")
ax.show()
```




```
#splittig x and y for predictoions

x=df.iloc[:,0:-1]
x
```

	age	balance	housing	loan	day	month	duration	pdays	previous
0	59	2343	1	0	5	8	1042	-1	0
1	56	45	0	0	5	8	1467	-1	0
2	41	1270	1	0	5	8	1389	-1	0
3	55	2476	1	0	5	8	579	-1	0
4	54	184	0	0	5	8	673	-1	0
...	...	...	...	...	...	...	...	...	...
11157	33	1	1	0	20	0	257	-1	0
11158	39	733	0	0	16	6	83	-1	0
11159	32	29	0	0	19	1	156	-1	0
11160	43	0	0	1	8	8	9	172	5

```
y=df["deposit"]
y
```

```
0        1
1        1
2        1
3        1
4        1
        ..
11157    0
11158    0
11159    0
11160    0
11161    0
Name: deposit, Length: 11162, dtype: int64
```

```
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.20,random_state=1)
```

```
lreg=LogisticRegression()
knn=KNeighborsClassifier()
svm=SVC()
dt=DecisionTreeClassifier()
```

```
def mymodel(model):
    model.fit(xtrain,ytrain)
    ypred=model.predict(xtest)
    ac=accuracy_score(ytest,ypred)
    cr=classification_report(ytest,ypred)
    print(f"accuracy-: {ac} \n classification report\n {cr}")
```

```
mymodel(lreg)
```

```
accuracy-: 0.7617554858934169
 classification report
              precision    recall  f1-score   support

           0       0.76      0.79      0.78      1165
           1       0.76      0.73      0.75      1068

    accuracy                           0.76      2233
   macro avg       0.76      0.76      0.76      2233
weighted avg       0.76      0.76      0.76      2233
```

```
mymodel(knn)
```

```
accuracy-: 0.7523510971786834
 classification report
              precision    recall  f1-score   support

           0       0.75      0.79      0.77      1165
           1       0.75      0.72      0.73      1068

    accuracy                           0.75      2233
   macro avg       0.75      0.75      0.75      2233
weighted avg       0.75      0.75      0.75      2233
```

```
mymodel(svm)
```

```
accuracy-: 0.7393640841916704
 classification report
              precision    recall  f1-score   support

           0       0.71      0.84      0.77      1165
           1       0.78      0.63      0.70      1068

    accuracy                           0.74      2233
   macro avg       0.75      0.73      0.73      2233
weighted avg       0.75      0.74      0.74      2233
```

```
mymodel(dt)
```

```
accuracy-: 0.7756381549484997
 classification report
```

```
                    precision    recall  f1-score   support

                 0       0.79      0.78      0.78      1165
                 1       0.76      0.78      0.77      1068

          accuracy                           0.78      2233
         macro avg       0.78      0.78      0.78      2233
      weighted avg       0.78      0.78      0.78      2233
```

```
#checking overfitted on dt
dt.score(xtrain,ytrain)
```

```
1.0
```

```
dt1=DecisionTreeClassifier(max_depth=5)
mymodel(dt1)
```

```
accuracy-: 0.7877295118674429
 classification report
                    precision    recall  f1-score   support

                 0       0.77      0.84      0.81      1165
                 1       0.81      0.73      0.77      1068

          accuracy                           0.79      2233
         macro avg       0.79      0.79      0.79      2233
      weighted avg       0.79      0.79      0.79      2233
```

```
for i in range(1,50):
 dt2=DecisionTreeClassifier(max_depth=i)
 dt2.fit(xtrain,ytrain)
 ypred=dt2.predict(xtest)
 print(f"{i}= {accuracy_score(ytest,ypred)}")
```

```
1= 0.7098074339453649
2= 0.7098074339453649
3= 0.7666815942678011
4= 0.7787729511867443
5= 0.7890729959695477
6= 0.7957904164800716
7= 0.8007165248544559
8= 0.8011643528884909
9= 0.8034034930586654
10= 0.8101209135691895
11= 0.8025078369905956
12= 0.7984773846842812
13= 0.800268696820421
14= 0.7980295566502463
15= 0.7948947604120018
16= 0.780564263322884
17= 0.7819077474249888
18= 0.780564263322884
19= 0.7769816390506046
20= 0.7760859829825347
21= 0.7693685624720108
22= 0.77384684281236
23= 0.77384684281236
24= 0.7747424988804299
25= 0.7747424988804299
26= 0.7729511867442902
27= 0.7711598746081505
28= 0.7671294223018361
29= 0.7720555306762203
30= 0.7792207792207793
31= 0.7778772951186744
32= 0.7756381549484997
33= 0.7765338110165696
34= 0.774294670846395
35= 0.7733990147783252
36= 0.77384684281236
37= 0.7689207344379758
38= 0.7693685624720108
39= 0.77384684281236
40= 0.768025078369906
41= 0.767577250335871
42= 0.7733990147783252
43= 0.77384684281236
44= 0.7725033587102552
```

```
45= 0.7733990147783252
46= 0.774294670846395
47= 0.77384684281236
48= 0.768025078369906
49= 0.7733990147783252
```

```
#best value of max_depth=10
dt3=DecisionTreeClassifier(max_depth=11)
mymodel(dt3)
```

```
accuracy-: 0.8034034930586654
 classification report
              precision    recall  f1-score   support

           0       0.81      0.81      0.81      1165
           1       0.79      0.80      0.80      1068

    accuracy                           0.80      2233
   macro avg       0.80      0.80      0.80      2233
weighted avg       0.80      0.80      0.80      2233
```

```
dt4=DecisionTreeClassifier(min_samples_leaf=10)
mymodel(dt4)
```

```
accuracy-: 0.8047469771607703
 classification report
              precision    recall  f1-score   support

           0       0.81      0.82      0.81      1165
           1       0.80      0.79      0.79      1068

    accuracy                           0.80      2233
   macro avg       0.80      0.80      0.80      2233
weighted avg       0.80      0.80      0.80      2233
```

```
for i in range(1,75):
 dt5=DecisionTreeClassifier(min_samples_leaf=i)
 dt5.fit(xtrain,ytrain)
 ypred=dt5.predict(xtest)
 print(f"{i}= {accuracy_score(ytest,ypred)}")
```

```
15= 0.8110165696372593
16= 0.8056426332288401
17= 0.8087774294670846
18= 0.8092252575011196
19= 0.8141513658755039
20= 0.8145991939095387
21= 0.8101209135691895
22= 0.8101209135691895
23= 0.8101209135691895
24= 0.8119122257053292
25= 0.8083296014330497
26= 0.8101209135691895
27= 0.8087774294670846
28= 0.8096730855351545
29= 0.8096730855351545
30= 0.8078817733990148
31= 0.8096730855351545
32= 0.8078817733990148
33= 0.8105687416032243
34= 0.8119122257053292
35= 0.8101209135691895
36= 0.8105687416032243
37= 0.8114643976712942
```

```
53= 0.8150470219435737
54= 0.8150470219435737
55= 0.8154948499776086
56= 0.8159426780116436
57= 0.8168383340797134
58= 0.8168383340797134
59= 0.8168383340797134
60= 0.819525302283923
61= 0.819525302283923
62= 0.819525302283923
63= 0.819525302283923
64= 0.819525302283923
65= 0.8154948499776086
66= 0.8159426780116436
67= 0.8092252575011196
68= 0.8092252575011196
69= 0.8132557098074339
70= 0.8132557098074339
71= 0.812807881773399
72= 0.812807881773399
73= 0.8105687416032243
```

```
dt6=DecisionTreeClassifier(min_samples_leaf=41)
mymodel(dt6)
```

```
     accuracy-: 0.8177339901477833
      classification report
                   precision    recall  f1-score   support

              0       0.85      0.79      0.82      1165
              1       0.79      0.84      0.82      1068

         accuracy                           0.82      2233
        macro avg       0.82      0.82      0.82      2233
     weighted avg       0.82      0.82      0.82      2233
```

```
dt7=DecisionTreeClassifier(max_depth=11,min_samples_leaf=41)
mymodel(dt7)
```

```
     accuracy-: 0.8159426780116436
      classification report
                   precision    recall  f1-score   support

              0       0.84      0.79      0.82      1165
              1       0.79      0.84      0.81      1068

         accuracy                           0.82      2233
        macro avg       0.82      0.82      0.82      2233
     weighted avg       0.82      0.82      0.82      2233
```

```
#gini
dt8=DecisionTreeClassifier(criterion="gini",min_samples_leaf=41)
mymodel(dt8)
```

```
     accuracy-: 0.8177339901477833
      classification report
                   precision    recall  f1-score   support

              0       0.85      0.79      0.82      1165
              1       0.79      0.84      0.82      1068

         accuracy                           0.82      2233
        macro avg       0.82      0.82      0.82      2233
     weighted avg       0.82      0.82      0.82      2233
```

```
dt9=DecisionTreeClassifier(criterion="gini",max_depth=10)
mymodel(dt9)
```

```
     accuracy-: 0.8105687416032243
      classification report
                   precision    recall  f1-score   support

              0       0.83      0.80      0.82      1165
              1       0.79      0.82      0.80      1068

         accuracy                           0.81      2233
        macro avg       0.81      0.81      0.81      2233
```

```
weighted avg       0.81      0.81      0.81      2233
```

```
dt10=DecisionTreeClassifier(criterion="entropy",max_depth=10)
mymodel(dt10)
```

```
accuracy-: 0.8150470219435737
 classification report
              precision    recall  f1-score   support

           0       0.81      0.84      0.83      1165
           1       0.82      0.79      0.80      1068

    accuracy                           0.82      2233
   macro avg       0.82      0.81      0.81      2233
weighted avg       0.82      0.82      0.81      2233
```

```
for i in range(1,50):
 dt11=DecisionTreeClassifier(criterion="entropy",max_depth=i)
 dt11.fit(xtrain,ytrain)
 ypred=dt11.predict(xtest)
 print(f"{i}= {accuracy_score(ytest,ypred)}")
```

```
1= 0.7098074339453649
2= 0.7098074339453649
3= 0.7532467532467533
4= 0.7711598746081505
5= 0.7823555754590238
6= 0.8007165248544559
7= 0.8096730855351545
8= 0.8150470219435737
9= 0.819525302283923
10= 0.8168383340797134
11= 0.8096730855351545
12= 0.8078817733990148
13= 0.7957904164800716
14= 0.7895208240035826
15= 0.7850425436632333
16= 0.7877295118674429
17= 0.7895208240035826
18= 0.7913121361397224
19= 0.793999104343932
20= 0.7877295118674429
21= 0.7850425436632333
22= 0.7850425436632333
23= 0.7877295118674429
24= 0.7854903716972682
25= 0.781012091356919
26= 0.786833855799373
27= 0.7836990595611285
28= 0.7886251679355127
29= 0.7863860277653381
30= 0.780564263322884
31= 0.7859381997313032
32= 0.7850425436632333
33= 0.786833855799373
34= 0.7836990595611285
35= 0.7890729959695477
36= 0.7774294670846394
37= 0.7859381997313032
38= 0.7823555754590238
39= 0.7854903716972682
40= 0.7828034034930587
41= 0.7823555754590238
42= 0.781012091356919
43= 0.7859381997313032
44= 0.7854903716972682
45= 0.7881773399014779
46= 0.7845947156291984
47= 0.7823555754590238
48= 0.786833855799373
49= 0.7819077474249888
```

```
#best value of max_depth when criterion=entropy
dt12=DecisionTreeClassifier(criterion="entropy",max_depth=10)
mymodel(dt12)
```

```
accuracy-: 0.8172861621137483
 classification report
```

```
                    precision    recall  f1-score   support

                0       0.82      0.84      0.83      1165
                1       0.82      0.79      0.81      1068

         accuracy                           0.82      2233
        macro avg       0.82      0.82      0.82      2233
     weighted avg       0.82      0.82      0.82      2233
```

```
dt13=DecisionTreeClassifier(criterion="entropy",min_samples_leaf=10)
mymodel(dt13)
```

```
     accuracy-: 0.8029556650246306
      classification report
                    precision    recall  f1-score   support

                0       0.81      0.81      0.81      1165
                1       0.79      0.80      0.79      1068

         accuracy                           0.80      2233
        macro avg       0.80      0.80      0.80      2233
     weighted avg       0.80      0.80      0.80      2233
```

```
for i in range(1,50):
 dt14=DecisionTreeClassifier(criterion="entropy",min_samples_leaf=i)
 dt14.fit(xtrain,ytrain)
 ypred=dt14.predict(xtest)
 print(f"{i}= {accuracy_score(ytest,ypred)}")
```

```
     1= 0.7881773399014779
     2= 0.7707120465741155
     3= 0.7845947156291984
     4= 0.7845947156291984
     5= 0.7890729959695477
     6= 0.793551276309897
     7= 0.799820868786386
     8= 0.8029556650246306
     9= 0.8092252575011196
     10= 0.8038513210927004
     11= 0.8056426332288401
     12= 0.8029556650246306
     13= 0.800268696820421
     14= 0.8007165248544559
     15= 0.8020600089565607
     16= 0.7984773846842812
     17= 0.8034034930586654
     18= 0.8016121809225257
     19= 0.8083296014330497
     20= 0.8110165696372593
     21= 0.8096730855351545
     22= 0.8132557098074339
     23= 0.8101209135691895
     24= 0.8083296014330497
     25= 0.8141513658755039
     26= 0.8150470219435737
     27= 0.8172861621137483
     28= 0.8163905060456784
     29= 0.8163905060456784
     30= 0.8163905060456784
     31= 0.8145991939095387
     32= 0.812807881773399
     33= 0.8101209135691895
     34= 0.8096730855351545
     35= 0.8105687416032243
     36= 0.8096730855351545
     37= 0.8074339453649798
     38= 0.8038513210927004
     39= 0.80653828929691
     40= 0.8092252575011196
     41= 0.8110165696372593
     42= 0.8141513658755039
     43= 0.8101209135691895
```

```
# best value of min_samples_leaf=15 when criterion=entropy
dt15=DecisionTreeClassifier(criterion="entropy",min_samples_leaf=15)
mymodel(dt15)
```

```
accuracy-: 0.8020600089565607
 classification report
              precision    recall  f1-score   support

           0       0.81      0.80      0.81      1165
           1       0.79      0.80      0.79      1068

    accuracy                           0.80      2233
   macro avg       0.80      0.80      0.80      2233
weighted avg       0.80      0.80      0.80      2233
```

**FINAL MODEL

```
#gini
dt16=DecisionTreeClassifier(max_depth=10,min_samples_leaf=41)
mymodel(dt16)
```

```
accuracy-: 0.8159426780116436
 classification report
              precision    recall  f1-score   support

           0       0.84      0.79      0.82      1165
           1       0.79      0.84      0.81      1068

    accuracy                           0.82      2233
   macro avg       0.82      0.82      0.82      2233
weighted avg       0.82      0.82      0.82      2233
```

```
#entropy
dt17=DecisionTreeClassifier(criterion="entropy",min_samples_leaf=15)
mymodel(dt17)
```

```
accuracy-: 0.8020600089565607
 classification report
              precision    recall  f1-score   support

           0       0.81      0.80      0.81      1165
           1       0.79      0.80      0.79      1068

    accuracy                           0.80      2233
   macro avg       0.80      0.80      0.80      2233
weighted avg       0.80      0.80      0.80      2233
```

```
# import Random Forest classifier

from sklearn.ensemble import RandomForestClassifier



# instantiate the classifier

rfc = RandomForestClassifier(random_state=0)



# fit the model

rfc.fit(xtrain, ytrain)



# Predict the Test set results

ypred = rfc.predict(xtest)



# Check accuracy score

from sklearn.metrics import accuracy_score
```