Lecture 12

May 30, 2023

1 Decision tree

Can be used for both classification and regression

1.1 Import The require libraries

[1]:
```
import numpy as np
import pandas as pd
```

[2]:
```
df=pd.read_csv('loan_prediction.csv')
```

[3]:
```
df.head()
```

[3]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	\
0	LP001002	Male	No	0	Graduate	No	
1	LP001003	Male	Yes	1	Graduate	No	
2	LP001005	Male	Yes	0	Graduate	Yes	
3	LP001006	Male	Yes	0	Not Graduate	No	
4	LP001008	Male	No	0	Graduate	No	

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	\
0	5849	0.0	NaN	360.0	
1	4583	1508.0	128.0	360.0	
2	3000	0.0	66.0	360.0	
3	2583	2358.0	120.0	360.0	
4	6000	0.0	141.0	360.0	

	Credit_History	Property_Area	Loan_Status
0	1.0	Urban	Y
1	1.0	Rural	N
2	1.0	Urban	Y
3	1.0	Urban	Y
4	1.0	Urban	Y

[4]:
```
df.shape
```

[4]:
```
(614, 13)
```

[5]:
```
df.describe()
```

[5]:
```
       ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
count       614.000000         614.000000  592.000000         600.00000
mean       5403.459283        1621.245798  146.412162         342.00000
std        6109.041673        2926.248369   85.587325          65.12041
min         150.000000           0.000000    9.000000          12.00000
25%        2877.500000           0.000000  100.000000         360.00000
50%        3812.500000        1188.500000  128.000000         360.00000
75%        5795.000000        2297.250000  168.000000         360.00000
max       81000.000000       41667.000000  700.000000         480.00000

       Credit_History
count      564.000000
mean         0.842199
std          0.364878
min          0.000000
25%          1.000000
50%          1.000000
75%          1.000000
max          1.000000
```

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

[6]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
```

```
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

[8]:
```
df.head()
```

[8]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

[9]:
```
# drop the unwanted columns
df.drop(columns=['Loan_ID','Gender','Dependents','Self_Employed'],inplace=True)
```

[10]:
```
df.head()
```

[10]:

	Married	Education	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	No	Graduate	5849	0.0	NaN	360.0	1.0	Urban	Y
1	Yes	Graduate	4583	1508.0	128.0	360.0	1.0	Rural	N
2	Yes	Graduate	3000	0.0	66.0	360.0	1.0	Urban	Y
3	Yes	Not Graduate	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	No	Graduate	6000	0.0	141.0	360.0	1.0	Urban	Y

```
   Loan_Amount_Term  Credit_History Property_Area Loan_Status
0             360.0             1.0         Urban           Y
1             360.0             1.0         Rural           N
2             360.0             1.0         Urban           Y
3             360.0             1.0         Urban           Y
4             360.0             1.0         Urban           Y
```

[11]:
```
#Handling Null Values
df['Married'].fillna('Yes',inplace=True)
df['LoanAmount'].fillna(df['LoanAmount'].mean(),inplace=True)
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mean(),inplace=True)
df['Credit_History'].fillna(df['Credit_History'].mean(),inplace=True)
```

[12]:
```
df.isnull().sum()
```

[12]:
```
Married              0
Education            0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
dtype: int64
```

[13]:
```
df.head()
```

[13]:
```
  Married     Education  ApplicantIncome  CoapplicantIncome  LoanAmount  \
0      No      Graduate             5849                0.0  146.412162
1     Yes      Graduate             4583             1508.0  128.000000
2     Yes      Graduate             3000                0.0   66.000000
3     Yes  Not Graduate             2583             2358.0  120.000000
4      No      Graduate             6000                0.0  141.000000

   Loan_Amount_Term  Credit_History Property_Area Loan_Status
0             360.0             1.0         Urban           Y
1             360.0             1.0         Rural           N
2             360.0             1.0         Urban           Y
3             360.0             1.0         Urban           Y
4             360.0             1.0         Urban           Y
```

[14]:
```
x=df.drop('Loan_Status',axis=1)
y=df['Loan_Status']
```

[15]:
```
x.shape
```

[15]: (614, 8)

[16]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
```

[17]:
```
ct=ColumnTransformer([('oh',OneHotEncoder(),[0,1,7])],remainder='passthrough')
```

[18]:
```
x=ct.fit_transform(x)
```

[19]:
```
x.shape
```

[19]: (614, 12)

[20]:
```
x
```

[20]:
```
array([[  1.        ,   0.        ,   1.        , …, 146.41216216,
         360.        ,   1.        ],
       [  0.        ,   1.        ,   1.        , …, 128.        ,
         360.        ,   1.        ],
       [  0.        ,   1.        ,   1.        , …,  66.        ,
         360.        ,   1.        ],
       …,
       [  0.        ,   1.        ,   1.        , …, 253.        ,
         360.        ,   1.        ],
       [  0.        ,   1.        ,   1.        , …, 187.        ,
         360.        ,   1.        ],
       [  1.        ,   0.        ,   1.        , …, 133.        ,
         360.        ,   0.        ]])
```

[21]:
```
from sklearn.preprocessing import LabelEncoder
```

[22]:
```
le=LabelEncoder()
```

[23]:
```
y=le.fit_transform(y)
```

[24]:
```
y
```

[24]:
```
array([1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0,
       0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1,
       1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1,
       1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1,
       0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
```

```
1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1,
1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0,
0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0,
1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0,
1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1,
1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1,
1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1,
1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1,
1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1,
0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0,
1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1,
1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0])
```

[25]:
```
from sklearn.preprocessing import StandardScaler
```

[26]:
```
sc=StandardScaler()
```

[27]:
```
x=sc.fit_transform(x)
```

[28]:
```
x
```

[28]:
```
array([[ 1.37208932e+00, -1.37208932e+00,  5.28362249e-01, …,
         3.38478577e-16,  2.79850543e-01,  4.51640451e-01],
       [-7.28815525e-01,  7.28815525e-01,  5.28362249e-01, …,
        -2.19273315e-01,  2.79850543e-01,  4.51640451e-01],
       [-7.28815525e-01,  7.28815525e-01,  5.28362249e-01, …,
        -9.57640999e-01,  2.79850543e-01,  4.51640451e-01],
       …,
       [-7.28815525e-01,  7.28815525e-01,  5.28362249e-01, …,
         1.26937121e+00,  2.79850543e-01,  4.51640451e-01],
       [-7.28815525e-01,  7.28815525e-01,  5.28362249e-01, …,
         4.83366900e-01,  2.79850543e-01,  4.51640451e-01],
       [ 1.37208932e+00, -1.37208932e+00,  5.28362249e-01, …,
        -1.59727534e-01,  2.79850543e-01, -2.41044061e+00]])
```

[29]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

[30]:
```
from sklearn.tree import DecisionTreeClassifier
```

[31]:
```
df=DecisionTreeClassifier(criterion='entropy',random_state=0)
```

[32]:
```
df.fit(x_train,y_train)
```

[32]:
```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

[33]:
```
pred=df.predict(x_test)
```

[34]:
```
pred
```

[34]:
```
array([0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
       0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1])
```

[35]:
```
y_test
```

[35]:
```
array([1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1,
       1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0,
       1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1])
```

[36]:
```
from sklearn.metrics import␣
↪accuracy_score,confusion_matrix,classification_report
```

[37]:
```
accuracy_score(y_test,pred)
```

[37]:
```
0.7235772357723578
```

[38]:
```
confusion_matrix(y_test,pred)
```

[38]:
```
array([[20, 13],
       [21, 69]])
```

[39]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.49      0.61      0.54        33
           1       0.84      0.77      0.80        90

    accuracy                           0.72       123
   macro avg       0.66      0.69      0.67       123
weighted avg       0.75      0.72      0.73       123
```