In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [4]:
```
df=pd.read_csv("cardio_train.csv.csv",delimiter=";")
```

In [5]:
```
df
```

Out[5]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	4	17474	1	156	56.0	100	60	1	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
69995	99993	19240	2	168	76.0	120	80	1	1	1	0	1	0
69996	99995	22601	1	158	126.0	140	90	2	2	0	0	1	1
69997	99996	19066	2	183	105.0	180	90	3	1	0	1	0	1
69998	99998	22431	1	163	72.0	135	80	1	2	0	0	0	1
69999	99999	20540	1	170	72.0	120	80	2	1	0	0	1	0

70000 rows × 13 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 13 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   id           70000 non-null  int64  
 1   age          70000 non-null  int64  
 2   gender       70000 non-null  int64  
 3   height       70000 non-null  int64  
 4   weight       70000 non-null  float64
 5   ap_hi        70000 non-null  int64  
 6   ap_lo        70000 non-null  int64  
 7   cholesterol  70000 non-null  int64  
 8   gluc         70000 non-null  int64  
 9   smoke        70000 non-null  int64  
 10  alco         70000 non-null  int64  
 11  active       70000 non-null  int64  
 12  cardio       70000 non-null  int64  
dtypes: float64(1), int64(12)
memory usage: 6.9 MB
```

data process

In [7]:
```
df.drop('id',axis=1,inplace=True)
```

In [8]:
```
df
```

Out[8]:

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	17474	1	156	56.0	100	60	1	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
69995	19240	2	168	76.0	120	80	1	1	1	0	1	0
69996	22601	1	158	126.0	140	90	2	2	0	0	1	1
69997	19066	2	183	105.0	180	90	3	1	0	1	0	1
69998	22431	1	163	72.0	135	80	1	2	0	0	0	1
69999	20540	1	170	72.0	120	80	2	1	0	0	1	0

70000 rows × 12 columns

In [9]:
```
df['age']=round(df['age']/365)
```

In [10]:
```
df
```

Out[10]:

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	50.0	2	168	62.0	110	80	1	1	0	0	1	0
1	55.0	1	156	85.0	140	90	3	1	0	0	1	1
2	52.0	1	165	64.0	130	70	3	1	0	0	0	1
3	48.0	2	169	82.0	150	100	1	1	0	0	1	1
4	48.0	1	156	56.0	100	60	1	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
69995	53.0	2	168	76.0	120	80	1	1	1	0	1	0
69996	62.0	1	158	126.0	140	90	2	2	0	0	1	1
69997	52.0	2	183	105.0	180	90	3	1	0	1	0	1
69998	61.0	1	163	72.0	135	80	1	2	0	0	0	1
69999	56.0	1	170	72.0	120	80	2	1	0	0	1	0

70000 rows × 12 columns

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   age          70000 non-null  float64
 1   gender       70000 non-null  int64  
 2   height       70000 non-null  int64  
 3   weight       70000 non-null  float64
 4   ap_hi        70000 non-null  int64  
 5   ap_lo        70000 non-null  int64  
 6   cholesterol  70000 non-null  int64  
 7   gluc         70000 non-null  int64  
 8   smoke        70000 non-null  int64  
 9   alco         70000 non-null  int64  
 10  active       70000 non-null  int64  
 11  cardio       70000 non-null  int64  
dtypes: float64(2), int64(10)
memory usage: 6.4 MB
```

In [12]:
```
df['age']=df['age'].astype('int64')
```

In [13]:
```
df
```

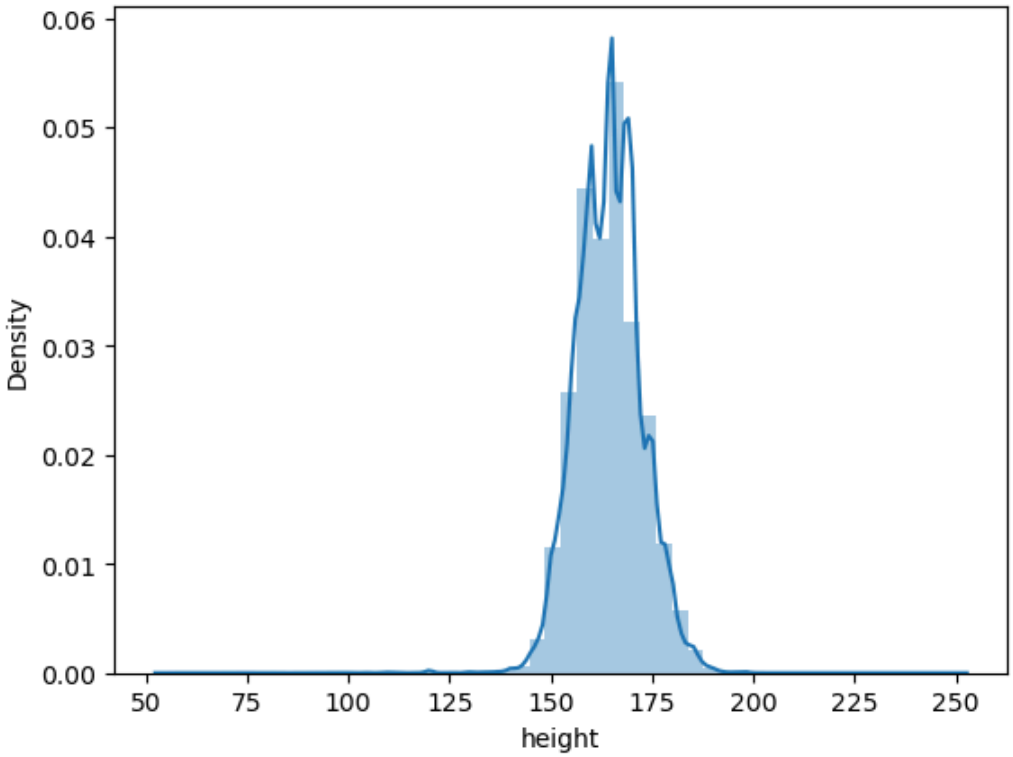
Out[13]:

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	50	2	168	62.0	110	80	1	1	0	0	1	0
1	55	1	156	85.0	140	90	3	1	0	0	1	1
2	52	1	165	64.0	130	70	3	1	0	0	0	1
3	48	2	169	82.0	150	100	1	1	0	0	1	1
4	48	1	156	56.0	100	60	1	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
69995	53	2	168	76.0	120	80	1	1	1	0	1	0
69996	62	1	158	126.0	140	90	2	2	0	0	1	1
69997	52	2	183	105.0	180	90	3	1	0	1	0	1
69998	61	1	163	72.0	135	80	1	2	0	0	0	1
69999	56	1	170	72.0	120	80	2	1	0	0	1	0

70000 rows × 12 columns

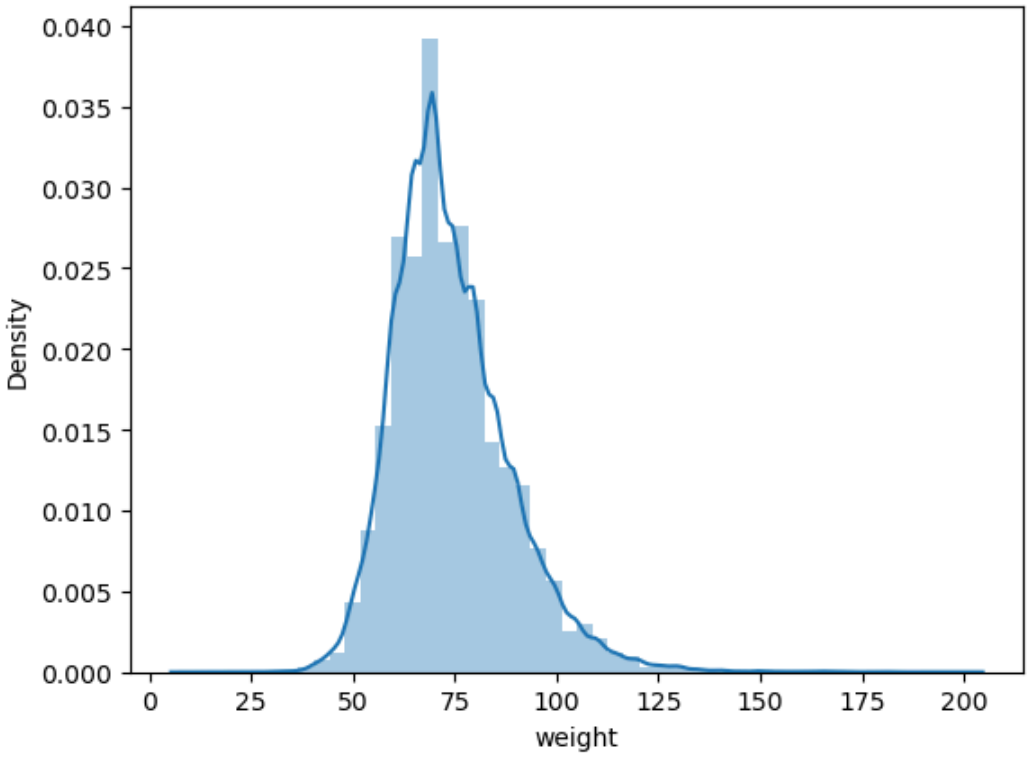
In [14]:
```
sns.distplot(df['height'])
```

Out[14]: <Axes: xlabel='height', ylabel='Density'>



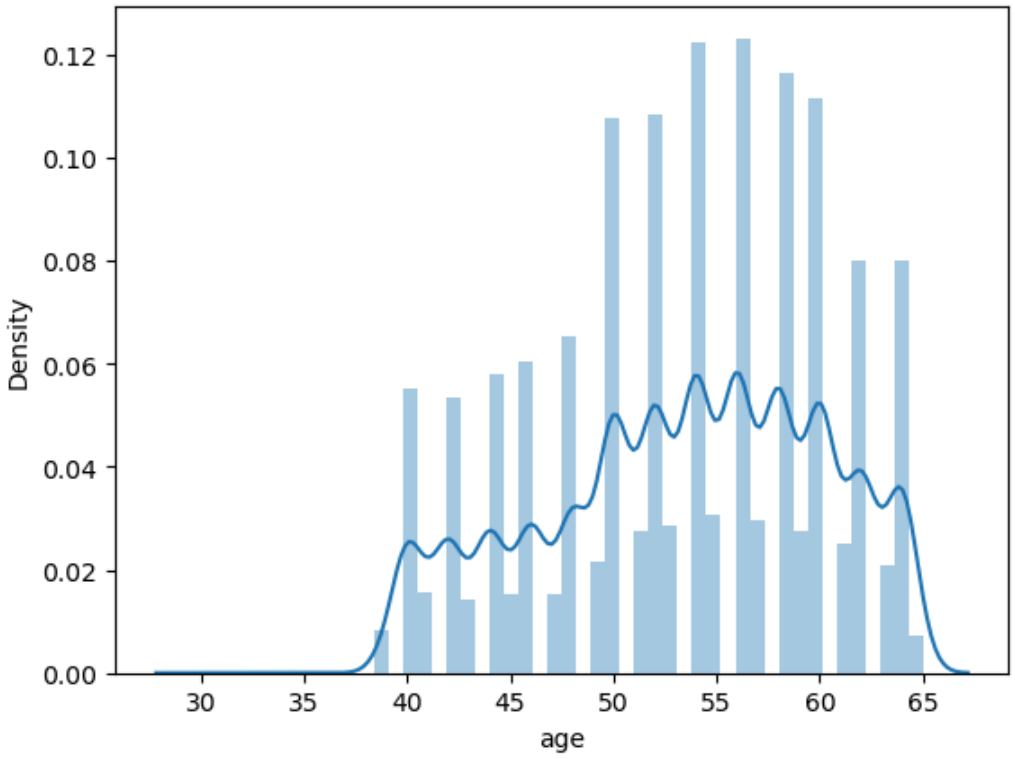
In [15]:
```
sns.distplot(df['weight'])
```

Out[15]: <Axes: xlabel='weight', ylabel='Density'>



In [16]:
```
sns.distplot(df['age'])
```

Out[16]: <Axes: xlabel='age', ylabel='Density'>



In [17]:
```
df[df['weight']<40];
```

In [18]:
```
df
```

Out[18]:

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	50	2	168	62.0	110	80	1	1	0	0	1	0
1	55	1	156	85.0	140	90	3	1	0	0	1	1
2	52	1	165	64.0	130	70	3	1	0	0	0	1
3	48	2	169	82.0	150	100	1	1	0	0	1	1
4	48	1	156	56.0	100	60	1	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...
69995	53	2	168	76.0	120	80	1	1	1	0	1	0
69996	62	1	158	126.0	140	90	2	2	0	0	1	1
69997	52	2	183	105.0	180	90	3	1	0	1	0	1
69998	61	1	163	72.0	135	80	1	2	0	0	0	1
69999	56	1	170	72.0	120	80	2	1	0	0	1	0

70000 rows × 12 columns

In [19]:
```
df['ap_lo'].unique()
```

Out[19]:
```
array([   80,    90,    70,   100,    60,    85,    89,   110,    65,
          63,    79,  1100,  1000,   800,   120,    50,    30,   109,
          84,  1033,   150,    91,    40,    73,    78,    75,    86,
          87,  1001,    82,    95,    69,    74,    97,    81,  1200,
          83,   119,     0,    93,   105, 10000,    99,    77,    59,
        8044,   140,    92,  1044,   108,   125,   115,    68,    61,
         106,   102,    94,    66,    52,   170,    76,   160,    62,
          96,   130,   113,    67,  9100,    10,    88,   902,     8,
         112,   104,    71,    72,  1008,    98,  2088,    20,   802,
        8000,  1022,   850,   708,    57,   101,  9011,  1011,    64,
        1007,  1177,  7100,    45,   709,  8500,    58,  1110,  8099,
        1088,   126,  1077,  1120,     7,   103,  1125,   180,   121,
        8100,   710,  5700,  8079,  1111,  1003,     6,  1900,   809,
         114,   801,  1002,    53,   111,     1,   118,    56,   182,
         810,     9,  7099, 11000,  9800,  8200,  1139,   107,   820,
          55,  1400,   190,   900,   122,  6800,   135,   700,    15,
        1101,   910,  1140,  1211,   -70,    54,  8077,   901,   880,
         870,   585,    49,   602], dtype=int64)
```

In [20]:
```
df['ap_hi'].unique()
```

Out[20]:
```
array([  110,   140,   130,   150,   100,   120,   145,   170,   135,
         125,    90,   180,   160,   133,   190,    80,   122,   169,
         126,   158,   200,    14,   123,    70,   161,   147,   115,
         137,   153,    11,   148,   105,   220,   119,   141,   165,
         164,    12,   124,   172,   902,   162,   906,   117,   134,
         166,   210,   176,   116,    10,   121,    16,   112,   159,
         113,   118,   155,   142,   131,   157,   136,   146,   138,
        -100,   909,   109,    85,   106,   129,    93,     7,    95,
         179,   156,   168,   132,   104,   103,   178,   175,   128,
         151,    15,   139, 11500,   127,    17,   108,   144,   102,
           1,  1420,    13,   143,   701,   107,   184,   149,   167,
         114,   101,    60,  1500,   181,   171,   202,  -115,   111,
         907,    20,   188,   185,   163,   173,   154,   177,  -140,
         174,  -120, 14020,  1400,   240,   191,   197,  1620,   152,
          96,   199,  -150,  1130,   193,    99,   196,   309,   401,
       16020,  1202,   806,  1300,   230,   207,   215,    97,  1409,
       11020,    24,   960, 13010,  1110,   195,  1205,   187,  2000],
      dtype=int64)
```

In [21]:
```
#deleting ap_lo less than 60 and greater than 150
df=df[df['ap_lo']>60]
df=df[df['ap_lo']<150]
```

In [22]:
```
#deleting ap_lo less than 250 and greater than 80
df=df[df['ap_hi']>80]
df=df[df['ap_hi']<250]
```

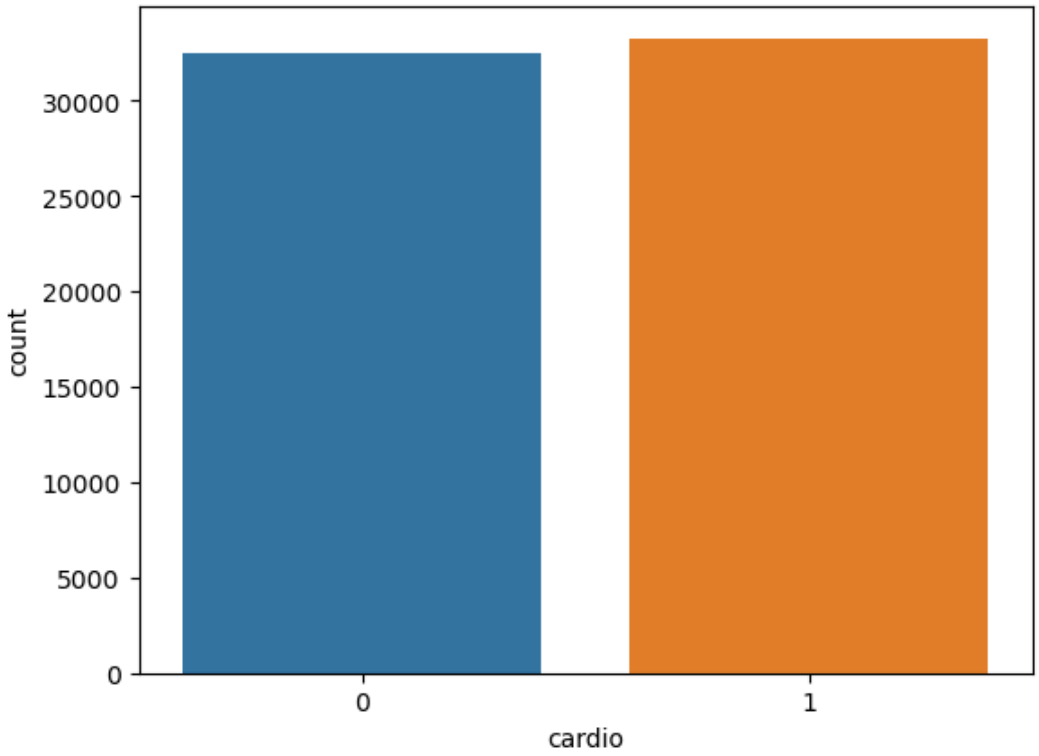
In [23]:
```
df=df[df['weight']>40]
```

In [24]:
```
#count of any column
df['cardio'].value_counts()
```

Out[24]:
```
1    33264
0    32521
Name: cardio, dtype: int64
```

In [25]:
```
sns.countplot(x='cardio',data=df)
```

Out[25]:
```
<Axes: xlabel='cardio', ylabel='count'>
```



splitting data

In [27]:
```
x=df.iloc[:,:-1]
y=df['cardio']
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.20,random_state=1)
```

In [31]:
```
from sklearn.linear_model import LogisticRegression
reg=LogisticRegression()
reg.fit(xtrain,ytrain)
```

Out[31]:
```
LogisticRegression
LogisticRegression()
```

In [33]:
```
y_train_pred=reg.predict(xtrain)
y_test_pred=reg.predict(xtest)
```

Evaluation of Model

In [37]:
```
from sklearn.metrics import accuracy_score
```

In [42]:
```
print("Traning data")
print(accuracy_score(ytrain,y_train_pred))


print("Testing data")
print(accuracy_score(ytest,y_test_pred))
```

```
Traning data
0.7117123964429581
Testing data
0.7122444326214182
```

In [44]:
```
from sklearn.metrics import confusion_matrix
print("Traning data")
print(confusion_matrix(ytrain,y_train_pred))
print()

print("Testing data")
print(confusion_matrix(ytest,y_test_pred))
```

```
Traning data
[[19471  6558]
 [ 8614 17985]]

Testing data
[[4863 1629]
 [2157 4508]]
```

In [47]:
```
from sklearn.metrics import classification_report,recall_score,precision_score
```

In [49]:
```
print("Traning data")
print(classification_report(ytrain,y_train_pred))
print()

print("Testing data")
print(classification_report(ytest,y_test_pred))
```

```
Traning data
              precision    recall  f1-score   support

           0       0.69      0.75      0.72     26029
           1       0.73      0.68      0.70     26599

    accuracy                           0.71     52628
   macro avg       0.71      0.71      0.71     52628
weighted avg       0.71      0.71      0.71     52628


Testing data
              precision    recall  f1-score   support

           0       0.69      0.75      0.72      6492
           1       0.73      0.68      0.70      6665

    accuracy                           0.71     13157
   macro avg       0.71      0.71      0.71     13157
weighted avg       0.71      0.71      0.71     13157
```

In [50]:
```
print(recall_score(ytrain,y_train_pred))
```

```
0.6761532388435656
```

In [52]:
```
y_train_prob=reg.predict_proba(xtrain)[:,1]
```

In [53]:
```
y_train_prob
```

Out[53]:
```
array([0.59567948, 0.32342783, 0.08975662, ..., 0.33057343, 0.49437176,
       0.64076902])
```

In [54]:
```
y_test_prob=reg.predict_proba(xtest)[:,1]
```

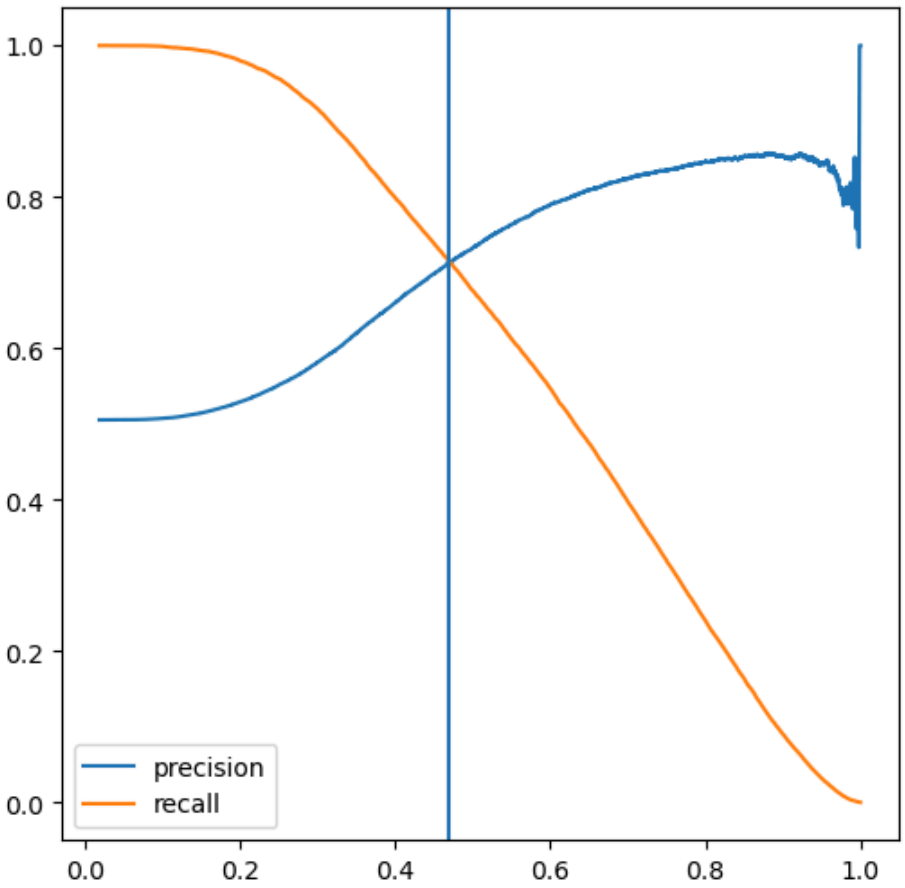
In [55]:
```
y_test_prob
```

Out[55]:
```
array([0.3969817 , 0.40896058, 0.43624011, ..., 0.40714681, 0.37728626,
       0.48620963])
```

In [57]:
```
from sklearn.metrics import precision_recall_curve
p,r,th=precision_recall_curve(ytrain,y_train_prob)
```

In [63]:
```
plt.figure(figsize=(6,6))
sns.lineplot(x=th,y=p[:-1],label='precision')
sns.lineplot(x=th,y=r[:-1],label='recall')
plt.axvline(0.47)
```

Out[63]:
```
<matplotlib.lines.Line2D at 0x1ff483f4be0>
```



In [*]:
```
#ROC_AUC_Curve
#reciver operator charateristics area under curve
from sklearn.metrics import roc_curve,auc
fpr,tpr,th=roc_curve(ytrain,y_train_prob)
sns.lineplot(x=fpr,y=tpr)
plt.axlable("FPR")
plt.ylable("TP")
```

In [ ]:
```

```