xgboost

# example code

|  |
| --- |
| # Import list  import xgboost as xgb  from xgboost import plot\_importance  import pandas as pd  import numpy as np  from sklearn.datasets import load\_breast\_cancer  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import confusion\_matrix, accuracy\_score, precision\_score, recall\_score  from sklearn.metrics import confusion\_matrix, f1\_score, roc\_auc\_score  import matplotlib.pyplot as plt  # Random Seed  SEED = 30  # Load Dataset from sklearn  dataset = load\_breast\_cancer()  X = dataset.data  Y = dataset.target  # Make DataFrame.  cancer\_df = pd.DataFrame(data=X,columns=dataset.feature\_names)  cancer\_df['target'] = Y  cancer\_df.head(3)  # Split data.  x\_train,x\_test,y\_train,y\_test = train\_test\_split(X,Y,test\_size=0.2,random\_state=SEED)  # Make train test DMatrix object (for XGBoost).  dtrain = xgb.DMatrix(data = x\_train, label= y\_train)  dtest = xgb.DMatrix(data=x\_test,label=y\_test)  # Setting Parameters.  params ={'max\_depth':3,  'eta':0.1,  'objective':'binary:logistic',  'eval\_metric':'logloss',  'early\_stoppings':100}  num\_rounds = 400  # Make model and train Model.  wlist = [(dtrain,'train'),(dtest,'eval')]  xgb\_model = xgb.train(params= params, dtrain= dtrain,num\_boost\_round=num\_rounds,evals=wlist)  # predict test data and check the result.  pred\_probs = xgb\_model.predict(dtest)  print("xgb\_moded Prediction Probability result (head(10))")  print(np.round(pred\_probs[:10],3))  preds = [1 if x>0.5 else 0 for x in pred\_probs]  print("xgb\_moded Prediction result (head(10)) (Probability > 0.5 => 1)")  print(preds[:10])  # def for model accuracy test.  def get\_accuracy(y\_test,y\_pred):  accuracy = accuracy\_score(y\_test,y\_pred)  precision = precision\_score(y\_test,y\_pred)  recall = recall\_score(y\_test,y\_pred)  F1 = f1\_score(y\_test,y\_pred)  AUC = roc\_auc\_score(y\_test,y\_pred)  print('\n accuarcy: {:.4f}'.format(accuracy))  print('precision: {:.4f}'.format(precision))  print('recall: {:.4f}'.format(recall))  print('F1: {:.4f}'.format(F1))  print('AUC: {:.4f}'.format(AUC))  # Model accuarcy test.  get\_accuracy(y\_test,preds)  # Draw Feature importance graph.  fig, ax = plt.subplots(figsize=(10,12))  plot\_importance(xgb\_model, ax=ax) |

# testing result

|  |
| --- |
| [0] train-logloss:0.60962 eval-logloss:0.61563  [1] train-logloss:0.54188 eval-logloss:0.55446  [2] train-logloss:0.48479 eval-logloss:0.50161  [3] train-logloss:0.43478 eval-logloss:0.45710  [4] train-logloss:0.39182 eval-logloss:0.41748  [5] train-logloss:0.35564 eval-logloss:0.38569  [6] train-logloss:0.32289 eval-logloss:0.35800  [7] train-logloss:0.29394 eval-logloss:0.33387  [8] train-logloss:0.26875 eval-logloss:0.31255  [9] train-logloss:0.24770 eval-logloss:0.29567  [10] train-logloss:0.22737 eval-logloss:0.27988  [11] train-logloss:0.21041 eval-logloss:0.26584  [12] train-logloss:0.19280 eval-logloss:0.25080  [13] train-logloss:0.17721 eval-logloss:0.23781  [14] train-logloss:0.16327 eval-logloss:0.22638  [15] train-logloss:0.15207 eval-logloss:0.21988  [16] train-logloss:0.14091 eval-logloss:0.21360  accuarcy: 0.9737  precision: 0.9718  recall: 0.9857  F1: 0.9787  AUC: 0.9701  <AxesSubplot:title={'center':'Feature importance'}, xlabel='F score', ylabel='Features'> |