from sklearn import model\_selection

from sklearn import metrics

import xgboost as xgb

from xgboost.sklearn import XGBClassifier

# read the data in

df = pd.read\_csv("E:/kfr/Diabetes.csv")

predictors = ['age','serum\_insulin']

target = 'class'

# Most common preprocessing step include label encoding and missing value treatment

from sklearn import preprocessing

for f in df.columns:

if df[f].dtype=='object':

lbl = preprocessing.LabelEncoder()

lbl.fit(list(df[f].values))

df[f] = lbl.transform(list(df[f].values))

df.fillna((-999), inplace=True)

# Let's use some week features to build the tree

X = df[['age','serum\_insulin']] # independent variables

y = df['class'].values # dependent variables

#Normalize

X = StandardScaler().fit\_transform(X)

# evaluate the model by splitting into train and test sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=2017)

num\_rounds = 100

kfold = model\_selection.StratifiedKFold(n\_splits=5, random\_state=2017, shuffle=True)

clf\_XGB = XGBClassifier(n\_estimators = num\_rounds,

objective= 'binary:logistic',

seed=2019)

# use early\_stopping\_rounds to stop the cv when there is no score imporovement

clf\_XGB.fit(X\_train,y\_train, early\_stopping\_rounds=20, eval\_set=[(X\_test, y\_test)], verbose=False)

results = model\_selection.cross\_val\_score(clf\_XGB, X\_train,y\_train, cv=kfold)

print ("\nxgBoost - CV Train : %.2f" % results.mean())

print ("xgBoost - Train : %.2f" % metrics.accuracy\_score(clf\_XGB.predict(X\_train), y\_train))

print ("xgBoost - Test : %.2f" % metrics.accuracy\_score(clf\_XGB.predict(X\_test), y\_test))

OUTPUT:

xgBoost - CV Train : 0.69

xgBoost - Train : 0.78

xgBoost - Test : 0.69