#Author @Bridge1998

**Import packages**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

from sklearn import datasets,model\_selection,linear\_model,metrics,decomposition

df = pd.read\_csv("UF\_R3\_ANALYSIS\_LASTMAJOR2007.csv")

df.isnull().sum()

df.dropna(axis=0, how='any',inplace=True)

df['STUDENT\_SUCCESSNO'].value\_counts()

**One Hot Encoding**

df\_JUNIOR\_SENIOR\_FLAGNO=df['JUNIOR\_SENIOR\_FLAGNO'].apply(str).str.get\_dummies().add\_prefix('JUNIOR\_SENIOR\_FLAGNO: ')

df\_ENRL\_CNT = df['ENRL\_CNT'].apply(str).str.get\_dummies().add\_prefix('ENRL\_CNT: ')

df\_MAJORCOUNT=df['MAJORCOUNT'].apply(str).str.get\_dummies().add\_prefix('MAJORCOUNT: ')

df\_UF\_CLASSNO=df['UF\_CLASSNO'].apply(str).str.get\_dummies().add\_prefix('UF\_CLASSNO: ')

df\_RESIDENCYNO=df['RESIDENCYNO'].apply(str).str.get\_dummies().add\_prefix('RESIDENCYNO: ')

df\_TERM\_END\_DT\_SID\_CATGRYNO=df['TERM\_END\_DT\_SID\_CATGRYNO'].apply(str).str.get\_dummies().add\_prefix('TERM\_END\_DT\_SID\_CATGRYNO: ')

df\_TERM\_BEG\_DT\_CATGRYNO=df['TERM\_BEG\_DT\_CATGRYNO'].apply(str).str.get\_dummies().add\_prefix('TERM\_BEG\_DT\_CATGRYNO: ')

df\_LOW\_TERM\_GPA\_IND=df['LOW\_TERM\_GPA\_IND'].apply(str).str.get\_dummies().add\_prefix('LOW\_TERM\_GPA\_IND: ')

df\_PARTTIME\_TERM\_IND=df['PARTTIME\_TERM\_IND'].apply(str).str.get\_dummies().add\_prefix('PARTTIME\_TERM\_IND: ')

df\_NOT\_REG\_TERM\_IND=df['NOT\_REG\_TERM\_IND'].apply(str).str.get\_dummies().add\_prefix('NOT\_REG\_TERM\_IND: ')

df\_WITHDRWL\_TERM\_IND=df['WITHDRWL\_TERM\_IND'].apply(str).str.get\_dummies().add\_prefix('WITHDRWL\_TERM\_IND: ')

df\_FULLTIME\_TERM\_IND=df['FULLTIME\_TERM\_IND'].apply(str).str.get\_dummies().add\_prefix('FULLTIME\_TERM\_IND: ')

df\_OVR\_12HR\_TERM\_IND=df['OVR\_12HR\_TERM\_IND'].apply(str).str.get\_dummies().add\_prefix('OVR\_12HR\_TERM\_IND: ')

df=pd.concat([df,df\_JUNIOR\_SENIOR\_FLAGNO,df\_ENRL\_CNT,df\_MAJORCOUNT,df\_UF\_CLASSNO,df\_RESIDENCYNO,df\_TERM\_END\_DT\_SID\_CATGRYNO,df\_TERM\_BEG\_DT\_CATGRYNO,df\_LOW\_TERM\_GPA\_IND,df\_PARTTIME\_TERM\_IND,df\_NOT\_REG\_TERM\_IND,df\_WITHDRWL\_TERM\_IND,df\_FULLTIME\_TERM\_IND,df\_OVR\_12HR\_TERM\_IND],axis=1)

df = df.drop([

'JUNIOR\_SENIOR\_FLAGNO',

'ENRL\_CNT',

'MAJORCOUNT',

'UF\_CLASSNO',

'RESIDENCYNO',

'TERM\_END\_DT\_SID\_CATGRYNO',

'TERM\_BEG\_DT\_CATGRYNO',

'LOW\_TERM\_GPA\_IND',

'PARTTIME\_TERM\_IND',

'NOT\_REG\_TERM\_IND',

'WITHDRWL\_TERM\_IND',

'FULLTIME\_TERM\_IND',

'OVR\_12HR\_TERM\_IND'

],axis=1)

**Create test dataset and train dataset**

df2 = pd .read\_csv("UF\_R3\_ANALYSIS\_LASTMAJOR0714BE.csv")

df2.to\_csv("train.csv", index=False)

df3 = df1[(df1['TERM\_BEG\_DT\_SID'] >=20150101) & (df1['TERM\_BEG\_DT\_SID'] <= 20191231)]

df3.to\_csv("test.csv", index=False)

train = pd.read\_csv("train.csv")

test = pd.read\_csv("test.csv")

mid = train['STUDENT\_SUCCESSNO']

train.drop(labels=['STUDENT\_SUCCESSNO'], axis=1,inplace = True)

train.insert(0, 'STUDENT\_SUCCESSNO', mid)

train\_x = train.values[0:, 1:]

train\_y = train.values[0:, 0]

result\_x = test.values[0:, 0:]

**Split the train dataset**

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test = train\_test\_split(train\_x, train\_y,test\_size=0.1,random\_state=33)

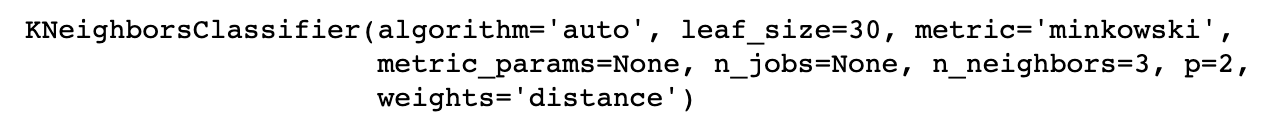
**Fit the model**

from numpy import ravel

from sklearn.neighbors import KNeighborsClassifier as kNN

model = kNN(n\_neighbors = 3, algorithm='auto', weights='distance')

model.fit(x\_train,ravel(y\_train))

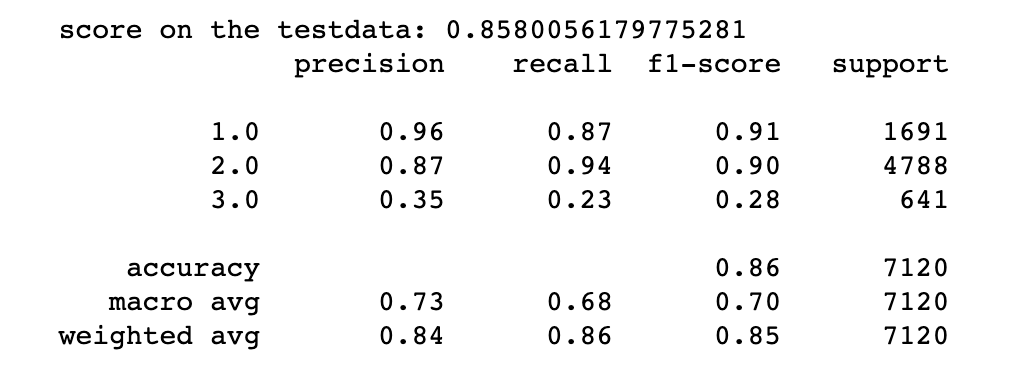


**Model result**

y\_predict = model.predict(x\_test)

print("score on the testdata:",model.score(x\_test,y\_test))

print(classification\_report(y\_test,y\_predict))



**List probability of predicting successfully, predicting results, and actual results. Create a csv file**

probablity = model.predict\_proba(x\_test)

list\_pro = []

for i in range(probablity.shape[0]):

pro = max(list(probablity[i]))

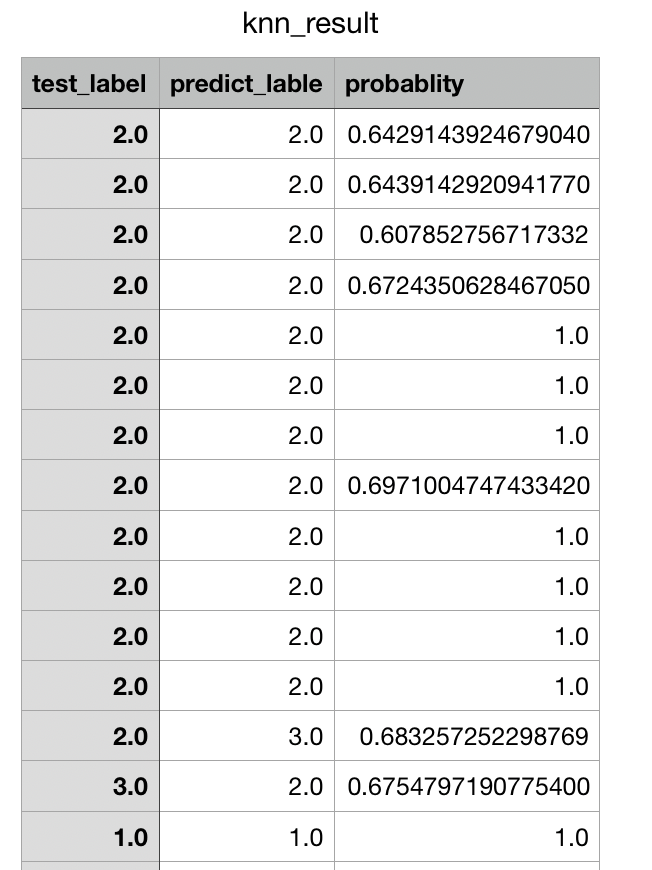
list\_pro.append(pro)

index = np.array(id).reshape((-1,1))[:,0:1]

result = pd.DataFrame(np.column\_stack((np.array(y\_test).reshape(-1,1),np.array(y\_predict).reshape(-1,1),np.array(list\_pro).reshape(-1,1))),

columns=['test\_label','predict\_lable','probablity'])

result.to\_csv('knn\_result.csv',index=False,header=True,encoding='gbk')



**Find all the incorrect results and create a csv file**

diff\_index = []

for i in range(result.shape[0]):

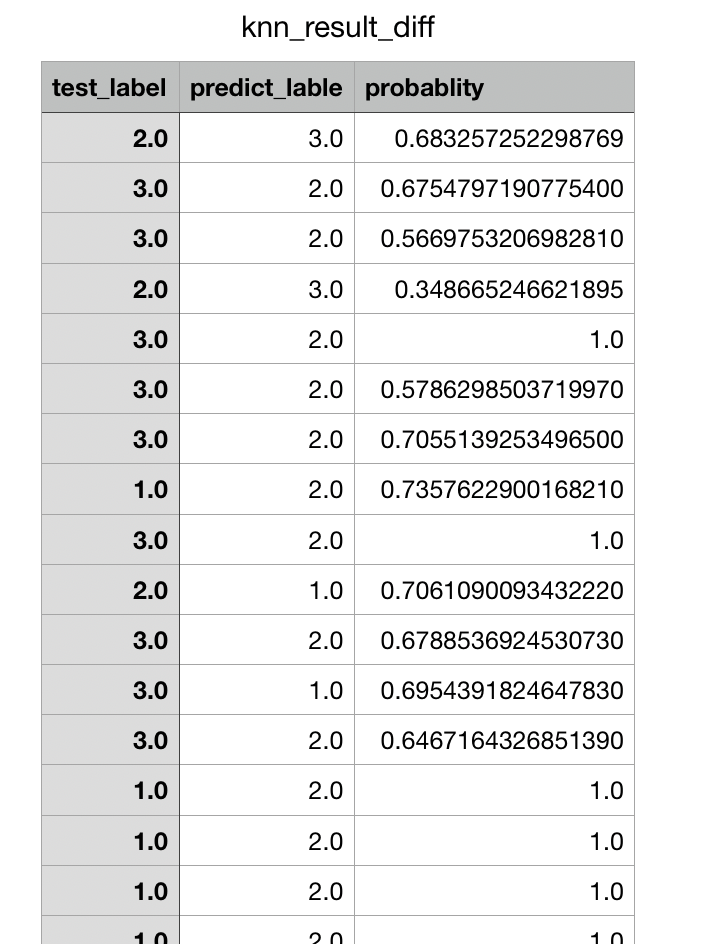
diff\_index.append(result['test\_label'][i] != result['predict\_lable'][i])

print(diff\_index)

diff = result[diff\_index]

diff\_x = x\_test[diff\_index]

diff.to\_csv('knn\_result\_diff.csv',index=False,header=True,encoding='gbk')



**Predict results of testing dataset and create a csv file**

mid = test['STUDENT\_SUCCESSNO']

test.drop(labels=['STUDENT\_SUCCESSNO'], axis=1,inplace = True)

test.insert(0, 'STUDENT\_SUCCESSNO', mid)

x\_test = test.values[0:, 1:]

y\_test = test.values[0:, 0]

y\_predict = model.predict(x\_test)

index = np.array(id).reshape((-1,1))[:,0:1]

result = pd.DataFrame(np.column\_stack(np.array(y\_predict).reshape(-1,1),)),

columns=['predict\_lable'])

result.to\_csv('predict\_result.csv',index=False,header=True,encoding='gbk')

