import pandas as pd

import numpy as np

df = pd.read\_excel('Quality evaluation of innovation and entrepreneurship education in colleges and universities.xlsx')

df.head(5)

df=df.drop(columns=['Serial number'])

df.head(5)

replace\_dict={'Strongly Disagree':1,'Disagree':2,'Neutral':3,'Agree':4,'Strongly Agree':5}

df=df.replace({'3、Curriculum Design—The resources, contents and materials of I&E are complete':replace\_dict,'3、The teaching method is unanimous':replace\_dict,'3、The training and practices of the I&E method is good':replace\_dict,'3、The contents of the curriculum have good relevance to professional courses':replace\_dict,'3、The curriculum is globally competitive':replace\_dict,

'4、Training and skill set enrichment—Proper and relevant training is incorporated as a professional activity':replace\_dict,'4、Training is given to students by well qualified trainers':replace\_dict,'5、Resources—Sufficient resources like smart rooms, workspaces, project development platforms, activity rooms etc are provided':replace\_dict,'5、Usage of Information and Communication Technology (ICT) tools':replace\_dict,'5、Financial policy and support to I&E activities':replace\_dict,

'6、Talent acquisition by teachers—Well qualified teachers':replace\_dict,'6、Development of I&E culture in teachers ':replace\_dict,'6、Enough number of teachers ':replace\_dict,'6、Availability of teachers in all disciplines':replace\_dict,'6、Experience and research capability of teachers ':replace\_dict,'7、Entrepreneurial environment—Constitution of entrepreneurial clubs ':replace\_dict,'7、Conduction of seminars, guest lectures, symposiums, contests, workshops, conferences and outreach activities.':replace\_dict})

df.head(5)

df\_Q1=df.iloc[:,0:5]

Q1=df\_Q1.sum(axis=1)/15

df\_Q1

df\_Q2=df.iloc[:,5:7]

Q2=df\_Q2.sum(axis=1)/15

df\_Q2

df\_Q3=df.iloc[:,7:10]

Q3=df\_Q3.sum(axis=1)/15

df\_Q3

df\_Q4=df.iloc[:,10:15]

Q4=df\_Q4.sum(axis=1)/15

df\_Q4

df\_Q5=df.iloc[:,15:17]

Q5=df\_Q5.sum(axis=1)/15

df\_Q5

# Feature engineering: normalization

from sklearn.preprocessing import StandardScaler

standardScaler1 = StandardScaler()

standardScaler1.fit(df\_Q1)

df\_Q1 = standardScaler1.transform(df\_Q1)

standardScaler2 = StandardScaler()

standardScaler2.fit(df\_Q2)

df\_Q2 = standardScaler2.transform(df\_Q2)

standardScaler3 = StandardScaler()

standardScaler3.fit(df\_Q3)

df\_Q3 = standardScaler3.transform(df\_Q3)

standardScaler4 = StandardScaler()

standardScaler4.fit(df\_Q4)

df\_Q4 = standardScaler4.transform(df\_Q4)

standardScaler5 = StandardScaler()

standardScaler5.fit(df\_Q5)

df\_Q5 = standardScaler5.transform(df\_Q5)

# Partitioning data sets

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

train\_X1,test\_X1,train\_y1,test\_y1 = train\_test\_split(df\_Q1,Q1,test\_size=0.2,random\_state=3)

train\_X2,test\_X2,train\_y2,test\_y2 = train\_test\_split(df\_Q2,Q2,test\_size=0.2,random\_state=3)

train\_X3,test\_X3,train\_y3,test\_y3 = train\_test\_split(df\_Q3,Q3,test\_size=0.2,random\_state=3)

train\_X4,test\_X4,train\_y4,test\_y4 = train\_test\_split(df\_Q4,Q4,test\_size=0.2,random\_state=3)

train\_X5,test\_X5,train\_y5,test\_y5 = train\_test\_split(df\_Q5,Q5,test\_size=0.2,random\_state=3)

# 【Random forest】

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report

from sklearn.metrics import accuracy\_score

rf1 = RandomForestClassifier(oob\_score=True)

rf1.fit(train\_X1,train\_y1.astype('string'))

rf\_pred\_y1 = rf1.predict(test\_X1)

print('Training set accuracy：

',rf1.score(train\_X1,train\_y1.astype('string')))

print('Test set accuracy：

',rf1.score(test\_X1,test\_y1.astype('string')))

print('Accuracy rate：

',accuracy\_score(test\_y1.astype('string'),rf\_pred\_y1))

print(classification\_report(test\_y1.astype('string'),rf\_pred\_y1))

rf2 = RandomForestClassifier(oob\_score=True)

rf2.fit(train\_X2,train\_y2.astype('string'))

rf\_pred\_y2 = rf2.predict(test\_X2)

print('Training set accuracy：

',rf2.score(train\_X2,train\_y2.astype('string')))

print('Test set accuracy：

',rf2.score(test\_X2,test\_y2.astype('string')))

print('Accuracy rate：

',accuracy\_score(test\_y2.astype('string'),rf\_pred\_y2))

print(classification\_report(test\_y2.astype('string'),rf\_pred\_y2))

rf3 = RandomForestClassifier(oob\_score=True)

rf3.fit(train\_X3,train\_y3.astype('string'))

rf\_pred\_y3 = rf3.predict(test\_X3)

print('Training set accuracy：

',rf3.score(train\_X3,train\_y3.astype('string')))

print('Test set accuracy：

',rf3.score(test\_X3,test\_y3.astype('string')))

print('Accuracy rate：

',accuracy\_score(test\_y3.astype('string'),rf\_pred\_y3))

print(classification\_report(test\_y3.astype('string'),rf\_pred\_y3))

rf4 = RandomForestClassifier(oob\_score=True)

rf4.fit(train\_X4,train\_y4.astype('string'))

rf\_pred\_y4 = rf4.predict(test\_X4)

print('Training set accuracy：

',rf4.score(train\_X4,train\_y4.astype('string')))

print('Test set accuracy：

',rf4.score(test\_X4,test\_y4.astype('string')))

print('Accuracy rate：

',accuracy\_score(test\_y4.astype('string'),rf\_pred\_y4))

print(classification\_report(test\_y4.astype('string'),rf\_pred\_y4))

rf5 = RandomForestClassifier(oob\_score=True)

rf5.fit(train\_X5,train\_y5.astype('string'))

rf\_pred\_y5 = rf5.predict(test\_X5)

print('Training set accuracy：

',rf5.score(train\_X5,train\_y5.astype('string')))

print('Test set accuracy：

',rf5.score(test\_X5,test\_y5.astype('string')))

print('Accuracy rate：

',accuracy\_score(test\_y5.astype('string'),rf\_pred\_y5))

print(classification\_report(test\_y5.astype('string'),rf\_pred\_y5))

df\_new=pd.concat([Q1,Q2,Q3,Q4,Q5],axis=1)

df\_new.rename(columns={0:'Q1',1:'Q2',2:'Q3',3:'Q4',4:'Q5'},inplace=True)

df\_new

TQE=df\_new.sum(axis=1)/5

TQE.loc[TQE>=0.6]=1

TQE.loc[TQE<0.6]=0

TQE

standardScaler = StandardScaler()

standardScaler.fit(df\_new)

df\_new = standardScaler.transform(df\_new)

train\_X,test\_X,train\_y,test\_y = train\_test\_split(df\_new,TQE,test\_size=0.2,random\_state=3)

# 【Logical regression model -- default parameters】

from sklearn.linear\_model import LogisticRegression

# predictor

log\_reg = LogisticRegression()

# Training model

log\_reg.fit(train\_X,train\_y)

# Forecast data

log\_pred\_y = log\_reg.predict(test\_X)

# Evaluation model

train\_score = log\_reg.score(train\_X,train\_y)

test\_score = log\_reg.score(test\_X,test\_y)

print('Training set accuracy：',train\_score)

print('Test set accuracy：',test\_score)

print(classification\_report(test\_y,log\_pred\_y))