****

****

**QG中期考核详细报告书**

**题    目     数据挖掘**

**学   院  自动化学院**

**专 业 数据科学与大数据技术**

**年级班别 20级2班**

**学 号 3120001548**

**学生姓名 钟逸圣**

**2021年 4 月 16 日**

**数据挖掘**

前期准备

1 调库及准备工作

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

​

%matplotlib inline

​

from IPython.core.interactiveshell import InteractiveShell

InteractiveShell.ast\_node\_interactivity = "all" # 一个cell显示多输出

​

pd.set\_option('display.max\_rows', 100) # 设置最大显示100行

pd.set\_option('display.max\_columns', 100) # 设置最大显示100列

2 调用数据及查看数据

2.1 调用数据及粗略查看数据

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

train\_data.head(5) #显示前5行数据

train\_data.tail(5) #显示后5行

train\_data.columns #查看列名

train\_data.info() #查看各字段的信息

train\_data.shape #查看数据集行列分布，几行几列

train\_data.describe() #查看数据的大体情况

2.2 查看数据缺失量及其占比

Ps：这步很关键，判断缺失值有哪些，从而选择如何处理

total = train\_data.isnull().sum().sort\_values(ascending=False)

print(total)

percent =(train\_data.isnull().sum()/train\_data.isnull().count()).sort\_values(ascending=False)

missing\_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])

missing\_data#.head(20)

2.3 查看数据类型

print(train\_data['difficulty\_level'].unique())

print(train\_data['education'].unique())

print(train\_data['total\_programs\_enrolled'].unique())

print(train\_data['city\_tier'].unique())

print(train\_data['is\_handicapped'].unique())

print(train\_data['age'].unique())

print(train\_data['trainee\_engagement\_rating'].unique())

print(train\_data['gender'].unique())

print(train\_data['program\_duration'].unique())

3 数据预处理

3.1热编码

train\_data.loc[train\_data['education'] == 'No Qualification', 'education'] = 0# 将education转换成值

train\_data.loc[train\_data['education'] == 'High School Diploma', 'education'] = 2

train\_data.loc[train\_data['education'] == 'Matriculation', 'education'] = 4

train\_data.loc[train\_data['education'] == 'Bachelors', 'education'] = 6

train\_data.loc[train\_data['education'] == 'Masters', 'education'] = 8

​

train\_data.loc[train\_data['difficulty\_level'] == 'easy', 'difficulty\_level'] = 0 #将level转换成值

train\_data.loc[train\_data['difficulty\_level'] == 'intermediate', 'difficulty\_level'] = 2

train\_data.loc[train\_data['difficulty\_level'] == 'hard', 'difficulty\_level'] = 4

train\_data.loc[train\_data['difficulty\_level'] == 'vary hard', 'difficulty\_level'] = 6

​

train\_data.loc[train\_data['test\_type'] == 'offline', 'test\_type'] = 0 #将test\_type转换成值

train\_data.loc[train\_data['test\_type'] == 'online', 'test\_type'] = 1

3.2 填充预览（先查看所有需填充数据的均值）

print(train\_data['age'].mean())

print(train\_data['trainee\_engagement\_rating'].mean())

print(train\_data['city\_tier'].mean())

print(train\_data['education'].mean())

print(train\_data['total\_programs\_enrolled'].mean())

print(train\_data['program\_duration'].mean())

print(train\_data['difficulty\_level'].mean())

3.3 填充及删除缺失行

train\_data['age'] = train\_data['age'].fillna(train\_data['age'].mean())

train\_data['gender'] = train\_data['gender'].fillna(train\_data['gender'].mean())

train\_data['trainee\_engagement\_rating'] = train\_data['trainee\_engagement\_rating'].fillna(train\_data['trainee\_engagement\_rating'].mean())

train\_data['city\_tier'] = train\_data['city\_tier'].fillna(train\_data['city\_tier'].mean())

train\_data['education'] = train\_data['education'].fillna(train\_data['education'].mean())

train\_data['test\_type'] = train\_data['test\_type'].fillna(train\_data['test\_type'].mode()[0])#线下较多

train\_data['total\_programs\_enrolled'] = train\_data['total\_programs\_enrolled'].fillna(train\_data['total\_programs\_enrolled'].mean())

train\_data['program\_duration'] = train\_data['program\_duration'].fillna(train\_data['program\_duration'].mean())

train\_data['is\_handicapped'] = train\_data['is\_handicapped'].fillna(0)#大部分人不残疾

train\_data['difficulty\_level'] = train\_data['difficulty\_level'].fillna(train\_data['difficulty\_level'].mean())

train\_data.dropna(inplace=True,axis=0,subset = ["program\_type","trainee\_id","test\_id","program\_id"])

4 处理后检查数据

4.1 查看数据是否有缺失

total = train\_data.isnull().sum().sort\_values(ascending=False)

print(total)

4.2 查看数据是否有未转换类型

print(train\_data['difficulty\_level'].unique())

print(train\_data['education'].unique())

print(train\_data['total\_programs\_enrolled'].unique())

print(train\_data['city\_tier'].unique())

print(train\_data['is\_handicapped'].unique())

print(train\_data['age'].unique())

print(train\_data['trainee\_engagement\_rating'].unique())

print(train\_data['gender'].unique())

print(train\_data['program\_duration'].unique())

特征工程

1 线性回归

1.1 调库及前期准备

from sklearn.linear\_model import LinearRegression # 导入线性回归的类

from sklearn.model\_selection import KFold # 交叉验证

predictors = ["trainee\_engagement\_rating","is\_handicapped", "total\_programs\_enrolled", "age", "gender", "difficulty\_level", "program\_duration", "city\_tier","education","test\_type"] # 输入机器学习算法的特征

alg = LinearRegression() # 初始化线性回归模型

alg1 = alg

kf = KFold(n\_splits=3,random\_state=2,shuffle=True) # KFold类实例化

​

predictions = []

MaxNum=0#最大拟合数

1.2 训练模型

for train, test in kf.split(train\_data):

train\_predictors = (train\_data[predictors].iloc[train,:]) # 取出训练数据，获得训练特征

train\_target = train\_data["is\_pass"].iloc[train] # 获取到数据集中交叉分类好的标签target，即是否通过

print(train\_predictors)

print(train\_target)

alg.fit(train\_predictors, train\_target) # 通过特征和target获得训练模型

print(train\_data[predictors].iloc[test,:])

test\_predictions = alg.predict(train\_data[predictors].iloc[test,:]) # 判断模型准确度

1.3 判断模型准确度

test\_predictions[test\_predictions>0.5]=1 #转换test\_predictions的值

test\_predictions[test\_predictions<=0.5]=0

print("预测结果：",test\_predictions)

print("真实结果：",test\_target)

print("正确的数量为：",sum(test\_predictions==test\_target))

print ("准确率为：",alg1.score(train\_data[predictors],train\_data['is\_pass']))

2 逻辑回归

from sklearn import model\_selection

from sklearn.linear\_model import LogisticRegression

​

alg = LogisticRegression(random\_state=1, solver='liblinear') # 初始化逻辑回归类

alg2 = alg

alg2.fit(train\_predictors, train\_target) # 通过特征和target获得训练模型

​

# 逻辑回归交叉验证

score = model\_selection.cross\_val\_score(alg2, train\_data[predictors], train\_data["is\_pass"], cv=3)

print("准确率为：", score.mean())

预测数据

1 测试数据导入

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

test.head(10)

total = test.isnull().sum().sort\_values(ascending=False)

print(total)

2 测试数据预处理

test.loc[test1['gender'] == 'F', 'gender'] = 0 # 令gender等于F那行的gender值为0

test.loc[test1['gender'] == 'M', 'gender'] = 1 # 令gender等于M那行的gender值为1

​

test.loc[test1['is\_handicapped'] == 'N', 'is\_handicapped'] = 0

test.loc[test1['is\_handicapped'] == 'Y', 'is\_handicapped'] = 1

​

test.loc[test1['education'] == 'No Qualification', 'education'] = 0# 将education转换成值

test.loc[test1['education'] == 'High School Diploma', 'education'] = 2

test.loc[test1['education'] == 'Matriculation', 'education'] = 4

test.loc[test1['education'] == 'Bachelors', 'education'] = 6

test.loc[test1['education'] == 'Masters', 'education'] = 8

​

test.loc[test1['difficulty\_level'] == 'easy', 'difficulty\_level'] = 0 #将level转换成值

test.loc[test1['difficulty\_level'] == 'intermediate', 'difficulty\_level'] = 2

test.loc[test1['difficulty\_level'] == 'hard', 'difficulty\_level'] = 4

test.loc[test1['difficulty\_level'] == 'vary hard', 'difficulty\_level'] = 6

test.loc[test1['test\_type'] == 'offline', 'test\_type'] = 0

test.loc[test1['test\_type'] == 'online', 'test\_type'] = 1

test['age'] = test['age'].fillna(test['age'].mean())

test['trainee\_engagement\_rating'] = test['trainee\_engagement\_rating'].fillna(test['trainee\_engagement\_rating'].mean())

test\_target=alg.predict(test1[predictors])

print(test\_target)

test\_target[test\_target>0.5]=1

test\_target[test\_target<=0.5]=0

print(test\_target)

3 测试数据导出

out=pd.read\_csv("./submission2.csv")#预览submission2

print (sub)

out['is\_pass']=test1\_target#将is\_pass数据赋值给submission2

sub.rename\_axis(index='')

print(out)

out.to\_csv("d:/QG/submission3.csv",index=None)#输出为文件submission3.csv

总结

数据挖掘其实就是将获取的数据集通过处理后利用其训练出一个能够近似预测结果的模型，其中包括数据预处理，构造特征工程，训练模型，预测结果几个步骤，而模型的准确度则表现在对数据的预处理方法以及对回归方法的选择。