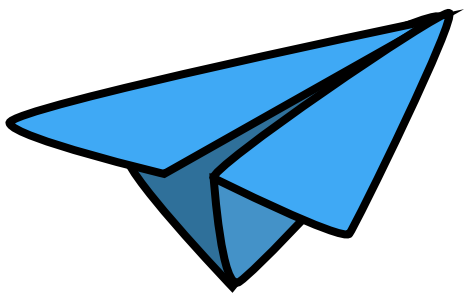
如何打一个数据挖掘比赛（进阶版）

贡献者名单

代码贡献

牧小熊、Datawhale成员



1. 前言

经过前面的练习，大家已经完成了相关的数据下载，并完了一个简单的模型构造，并提交了相关果，如果你还没有完成，可以先学习 如何打一个数据挖掘比赛V2.0 ，然后再进行本节的学习。

本次比赛是一个医疗领域数据挖掘赛，需要选手通过训练集数据构建模型，对验证集数据进行预测，并将预测的结果提交到科大讯飞数据竞赛平台中。

2. 提问与反馈

如果在学习过程中遇到更多问题，可在Q&A文档中记录，我们会跟进反馈

更新时间：2022-07-14

访问链接：糖尿病遗传风险挑战赛问题反馈

3. 教程内容

主线任务需要学习者独立完成

支线任务为学有余力的同学独立完成

思考为学习者可以思考的方向，可通过讨论或搜索获得结果

比赛地址：https://challenge.xfyun.cn/topic/info?type=diabetes&ch=ds22-dw-wd05

任务名称	难度	备注
任务1：比赛报名与环境配置	⭐	报名是数据挖掘比赛中的最重要的一步，在这一环节中需要获取比赛数据，同时获得比赛的相关信息，例如：比赛时间、数评价指标等
任务2：数据的读取与数据类型	⭐	查看数据的类型，数据的大小，为任务3做准备
任务3：数据的分析与探索	⭐	数据探索是任务2基础上进行进一步的分析数据类型，包括数据的缺失值，异常值，文本类型值，数据的分布，数据间的相关性
任务4：数据的特征工程	⭐⭐	数据挖掘中的最重要的步骤，通过不同特征数据值的构造，来挖掘出更高关联的特征
任务5：模型的构建	⭐⭐⭐	数据挖掘比赛中，模型的构建对应的是将数据向预测的一个转换过程，好的模型的选择能大幅度提升模型预测准确率。
任务6：模型构建的进阶	⭐⭐⭐⭐⭐	模型后续进阶的不同方法介绍

任务1：比赛报名与环境配置

主线任务：

1. 访问糖尿病遗传风险检测挑战赛网页，并注册相关账号
2. 点击页面中 `赛事概要` ，了解比赛的赛事背景、赛事任务、提交说明、评估指标等相关信息
3. 安装并配置好 `python` 的编程环境

思考：

1. 为什么要了解比赛的相关信息？
2. 比赛的评估指标有哪几种？本次比赛中为什么使用F1-score，相比其他评估指标有什么优势？

任务2：数据的读取与数据类型

主线任务：

1. 解压比赛数据，使用 `panda` 读取比赛数据，并查看训练集和测试集数据大小
2. 查看训练集和测试集的数据类型

思考：

1. 为什么要查看训练集和测试集的大小？
2. 为什么查看训练集和测试集的数据类型？

参考代码：

```
import pandas as pd
train_df=pd.read_csv('比赛训练集.csv',encoding='gbk')
test_df=pd.read_csv('比赛测试集.csv',encoding='gbk')

print('训练集的数据大小：',train_df.shape)
print('测试集的数据大小：',test_df.shape)
print('-'*30)
print('训练集的数据类型：')
print(train_df.dtypes)
print('-'*30)
print(test_df.dtypes)
```

任务3：数据的分析与探索

主线任务：

1. 查看训练集和测试集的缺失值，并比训练集和测试集的缺失值分布是否一致
2. 使用 `.corr()` 函数查看数据间的相关性
3. 对训练集和测试集数据进行可视化统计

思考：

1. 数据中的缺失值产生的原因？

2. 怎么查看数据间的相关性？如果相关性高说明了什么？

参考代码：

```
1
2  #----------------查数据的缺失值----------------
3  print(train_df.isnull().sum())
4  print('-'*30)
5  print(test_df.isnull().sum())
6  #可以看到 训练集和测试集中都是舒张压有缺失值
7
8  #----------------查数据相关性----------------
9  print('-'*30)
10 print('查看训练集中数据的相关性')
11 print(train_df.corr())
12 print(test_df.corr())
13 #----------------数据的可视化统计----------------
14 import matplotlib.pyplot as plt
15 import seaborn as sns
16
17 train_df['性别'].value_counts().plot(kind='barh')
18 sns.set(font='SimHei',font_scale=1.1)  # 解决Seaborn中文显示问题并调整字体大小
19 sns.countplot(x='患有糖尿病标识', hue='性别', data=train_df)
20 sns.boxplot(y='出生年份', x='患有糖尿病标识', hue='性别', data=train_df)
21 sns.violinplot(y='体重指数', x='患有糖尿病标识', hue='性别', data=train_df)
22
23 plt.figure(figsize = [20,10],dpi=100)
24 sns.countplot(x='出生年份',data=train_df)
25 plt.tight_layout()
```

任务4：数据的特征工程

主线任务：

1. 将数据中的糖尿病家族史中的文本数据进行编码
2. 将数据中的舒张压的缺失值进行填充
3. 将出生年份的数据转换成年龄数据并进行分组
4. 对体重和舒张压的数据进行分组
5. 删除数据中的编号这一列

支线任务：

1. 计算每个个体口服耐糖量测试、胰岛素释放实验、舒张压这三个指标对糖尿病家族史进行分组求平均值后的差值
2. 计算每个个体口服耐糖量测试、胰岛素释放实验、舒张压这三个指标对年龄进行分组求平均值后的差值

思考：

1. 文本数据为什么要进行编码?有没有其他的处理方法？除了编码为连续数字，有没有其他形式?
2. 为什么要填充缺失值？你觉得参考代码中将所有的缺失值全部填充为0是否正确?
3. 为什么要将出生年份转换成年龄？为什么要对年龄分组?
4. 为什么对体重和舒张压进行了分组？这么做是否正确?
5. 为什么要删除编号这一列?

参考代码：

```
#这里将文本数据转成数字数据
dict_糖尿病家族史 = {
    '无记录': 0,
    '叔叔或姑姑有一方患有糖尿病': 1,
    '叔叔或者姑姑有一方患有糖尿病': 1,
    '父母有一方患有糖尿病': 2
}

train_df['糖尿病家族史'] = train_df['糖尿病家族史'].map(dict_糖尿病家族史)
test_df['糖尿病家族史'] = test_df['糖尿病家族史'].map(dict_糖尿病家族史)

#考虑到舒张压是一个较为重要的生理特征，并不能适用于填充平均值，这里采用填充为0的方法
train_df['舒张压'].fillna(0, inplace=True)
test_df['舒张压'].fillna(0, inplace=True)

#将数据中的出生年份换算成年龄
train_df['出生年份'] = 2022 - train_df['出生年份']
test_df['出生年份'] = 2022 - test_df['出生年份']

#将年龄进行一个分类
"""
>50
<=18
19-30
31-50
"""
def resetAge(input):
    if input<=18:
        return 0
    elif 19<=input<=30:
        return 1
    elif 31<=input<=50:
        return 2
    elif input>=51:
        return 3

```

```
train_df['rAge']=train_df['出生年份'].apply(resetAge)
test_df['rAge']=test_df['出生年份'].apply(resetAge)

#将体重指数进行一个分类
"""
人体的成人体重指数正常值是在18.5-24之间
低于18.5是体重指数过轻
在24-27之间是体重超重
27以上考虑是肥胖
高于32了就是非常的肥胖。
"""
def BMI(a):
    if a<18.5:
        return 0
    elif 18.5<=a<=24:
        return 1
    elif 24<a<=27:
        return 2
    elif 27<a<=32:
        return 3
    else:
        return 4

train_df['BMI']=train_df['体重指数'].apply(BMI)
test_df['BMI']=test_df['体重指数'].apply(BMI)
#将舒张压进行一个分组
"""
舒张压范围为60-90
"""
def DBP(a):
    if a==0:#这里为数据缺失的情况
        return 0
    elif 0<a<60:
        return 1
    elif 60<=a<=90:
        return 2
    else:
        return 3
train_df['DBP']=train_df['舒张压'].apply(DBP)
test_df['DBP']=test_df['舒张压'].apply(DBP)

#删除编号
train_df=train_df.drop(['编号'],axis=1)
test_df=test_df.drop(['编号'],axis=1)
```

```
#以下是支线任务参考代码

#这里计算口服耐糖量相对糖尿病家族史进行分组求平均值后的差值
train_df['口服耐糖量测试_diff'] = abs(train_df['口服耐糖量测试'] - train_df.groupby('糖尿病家族史').transform('mean')['口服耐糖量测试'])
test_df['口服耐糖量测试_diff'] = abs(test_df['口服耐糖量测试'] - test_df.groupby('糖尿病家族史').transform('mean')['口服耐糖量测试'])

#这里计算口服耐糖量相对年龄进行分组求平均值后的差值
train_df['口服耐糖量测试_diff'] = abs(train_df['口服耐糖量测试'] - train_df.groupby('rAge').transform('mean')['口服耐糖量测试'])
test_df['口服耐糖量测试_diff'] = abs(test_df['口服耐糖量测试'] - test_df.groupby('rAge').transform('mean')['口服耐糖量测试'])

```

任务5：模型的构建与优化

主线任务：

1. 构建用于模型训练的训练集、训练标签以及测试集
2. 从以下4个不同模型中选择1个完成模型构建，并提交分数

思考：

1. 能够用于二分类的机器学习算法有哪些？
2. 在逻辑回归代码中，为什么要进行数据标准化？
3. 本次比赛中逻辑回归算法有较差的分数可能有哪些原因？

参考代码：

```
train_label=train_df['患有糖尿病标识']
train=train_df.drop(['患有糖尿病标识'],axis=1)
test=test_df
```

逻辑回归（分数：0.74）：

```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import make_pipeline

# 构建模型
model = make_pipeline(
    MinMaxScaler(),
```

```
    LogisticRegression()
)
model.fit(train,train_label)
pre_y=model.predict(test)
result=pd.read_csv('提交示例.csv')
result['label']=pre_y
result.to_csv('LR.csv',index=False)
```

决策树（分数：0.93）：

```
from sklearn.tree import DecisionTreeClassifier

# 构建模型
model = DecisionTreeClassifier()
model.fit(train,train_label)
pre_y=model.predict(test)
result=pd.read_csv('提交示例.csv')
result['label']=pre_y
result.to_csv('CART.csv',index=False)
```

lightgbm版本（分数：0.95）：

```
import lightgbm
def select_by_lgb(train_data,train_label,test_data,random_state=2022,metric='auc',num_round=300):
    clf=lightgbm
    train_matrix=clf.Dataset(train_data,label=train_label)

    params={
            'boosting_type': 'gbdt',
            'objective': 'binary',
            'learning_rate': 0.1,
            'metric': metric,
            'seed': 2020,
            'nthread':-1 }
    model=clf.train(params,train_matrix,num_round)
    pre_y=model.predict(test_data)
    return pre_y

#输出预测值
test_data=select_by_lgb(train,train_label,test)
pre_y=pd.DataFrame(test_data)
pre_y['label']=pre_y[0].apply(lambda x:1 if x>0.5 else 0)
result=pd.read_csv('提交示例.csv')
```

```
result['label']=pre_y['label']
result.to_csv('lgb.csv',index=False)
```

lightgbm版本5折交叉验证（分数：0.96）：

```
import lightgbm
from sklearn.model_selection import KFold
def select_by_lgb(train_data,train_label,test_data,random_state=2022,n_splits=5,metric='auc',num_round=10000,early_stopping_rounds=100):
    kfold = KFold(n_splits=n_splits, shuffle=True, random_state=random_state)
    fold=0
    result=[]
    for train_idx, val_idx in kfold.split(train_data):
        random_state+=1
        train_x = train_data.loc[train_idx]
        train_y = train_label.loc[train_idx]
        test_x = train_data.loc[val_idx]
        test_y = train_label.loc[val_idx]
        clf=lightgbm
        train_matrix=clf.Dataset(train_x,label=train_y)
        test_matrix=clf.Dataset(test_x,label=test_y)
        params={
                'boosting_type': 'gbdt',
                'objective': 'binary',
                'learning_rate': 0.1,
                'metric': metric,
                'seed': 2020,
                'nthread':-1 }
        model=clf.train(params,train_matrix,num_round,valid_sets=test_matrix,early_stopping_rounds=early_stopping_rounds)
        pre_y=model.predict(test_data)
        result.append(pre_y)
        fold+=1
    return result

test_data=select_by_lgb(train,train_label,test)
pre_y=pd.DataFrame(test_data).T
pre_y['averge']=pre_y[[i for i in range(5)]].mean(axis=1)
pre_y['label']=pre_y['averge'].apply(lambda x:1 if x>0.5 else 0)
result=pd.read_csv('提交示例.csv')
result['label']=pre_y['label']
result.to_csv('lgb.csv',index=False)
```

任务6：模型构建的进阶：

主线任务：

1. 使用不同模型来评估预测准确性
2. 对3个预测准确度最高的模型参数的搜索，并比较不同模型的预测准确性

思考：

1. 模型融合的优点在哪里？
2. 运行主线任务1，思考这些算法为什么要较高的准确度？
3. 为什么可以通过搜索来调整模型的参数？模型参数的调整一定会让预测更准确嘛？
4. 你觉得参考代码中搜索的参数设置合理嘛？如果不合理应该如何改进？

参考代码：

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn.linear_model import LogisticRegression


train_label=train_df['患有糖尿病标识']
train=train_df.drop(['患有糖尿病标识'],axis=1)
test=test_df
#分割训练集和验证集
train_x,val_x,train_y,val_y=train_test_split(train,train_label,test_size=0.25,random_state=2020)
model={}
model['rfc']=RandomForestClassifier()
model['gdbt']=GradientBoostingClassifier()
model['cart']=DecisionTreeClassifier()
model['knn']=KNeighborsClassifier()
model['svm']=svm.SVC()
model['lr']=LogisticRegression()
for i in model:
    model[i].fit(train_x,train_y)
    score=cross_val_score(model[i],val_x,val_y,cv=5,scoring='f1')
    print('%s的f1为：%.3f'%(i,score.mean()))

"""
rfc的f1为：0.927
gdbt的f1为：0.925
cart的f1为：0.899
knn的f1为：0.811
svm的f1为：0.751
lr的f1为：0.718
"""
```

```
from sklearn.model_selection import GridSearchCV

model=['rfc','gbdt','cart']

temp=[]
rfc=RandomForestClassifier(random_state=0)
params={'n_estimators':[50,100,150,200,250],'max_depth':[1,3,5,7,9,11,13,15,17,19],'min_samples_leaf':[2,4,6]}
temp.append([rfc,params])

gbt=GradientBoostingClassifier(random_state=0)
params={'learning_rate':[0.01,0.05,0.1,0.15,0.2],'n_estimators':[100,300,500],'max_depth':[3,5,7]}
temp.append([gbt,params])

cart=DecisionTreeClassifier(random_state=0)
params={'max_depth':[1,3,5,7,9,11,13,15,17,19],'min_samples_leaf':[2,4,6]}
temp.append([cart,params])

for i in range(len(model)):
    best_model=GridSearchCV(temp[i][0],param_grid=temp[i][1],refit=True,cv=5).fit(train,train_label)
    print(model[i],':')
    print('best parameters:',best_model.best_params_)

"""
rfc :
best parameters: {'max_depth': 17, 'min_samples_leaf': 2, 'n_estimators': 100}
gbdt :
best parameters: {'learning_rate': 0.01, 'max_depth': 7, 'n_estimators': 300}
cart :
best parameters: {'max_depth': 7, 'min_samples_leaf': 2}
"""

model={}
model['rfc']=RandomForestClassifier(max_depth=17,min_samples_leaf=2,n_estimators=100)
model['gdbt']=GradientBoostingClassifier(learning_rate=0.01,max_depth=7,n_estimators=300)
model['cart']=DecisionTreeClassifier(max_depth=7,min_samples_leaf=2)
for i in model:
    model[i].fit(train_x,train_y)
    score=cross_val_score(model[i],val_x,val_y,cv=5,scoring='f1')
    print('%s的f1为: %.3f'%(i,score.mean()))

"""
rfc的f1为: 0.931
gdbt的f1为: 0.922
```

```
cart的f1为：0.920
"""
```

---

rfc版本（分数：0.965）：

```
model=RandomForestClassifier(max_depth=17,min_samples_leaf=2,n_estimators=100)
model.fit(train,train_label)
pre_y=model.predict(test)
result=pd.read_csv('提交示例.csv')
result['label']=pre_y
result.to_csv('rfc.csv',index=False)
```

4.后续

行文至此，数据挖掘比赛项目就告一段落了，经过这2次教程的学习，你应该体验到了数据挖掘比赛从报名到模型构建到优化的全过程，这将是你打开数据科学/算法工程/数据分析的第一步。正所谓“路漫漫其修远兮，吾将上下而求索”，这一步终究只是开始，在距离你的成为AI大师还有漫长的路要探索，但这也是一个美好的开始。正所谓“千里之行，始于足下”，相信这个简短的数据挖掘比赛教程将打开你数据挖掘的大门，若干年后，你将还会记得当初那个跟着教程不断尝试的自己。也期待成长后你加入幕后的贡献者团队，我们将一起坚持初心，帮助更多学习者成长。