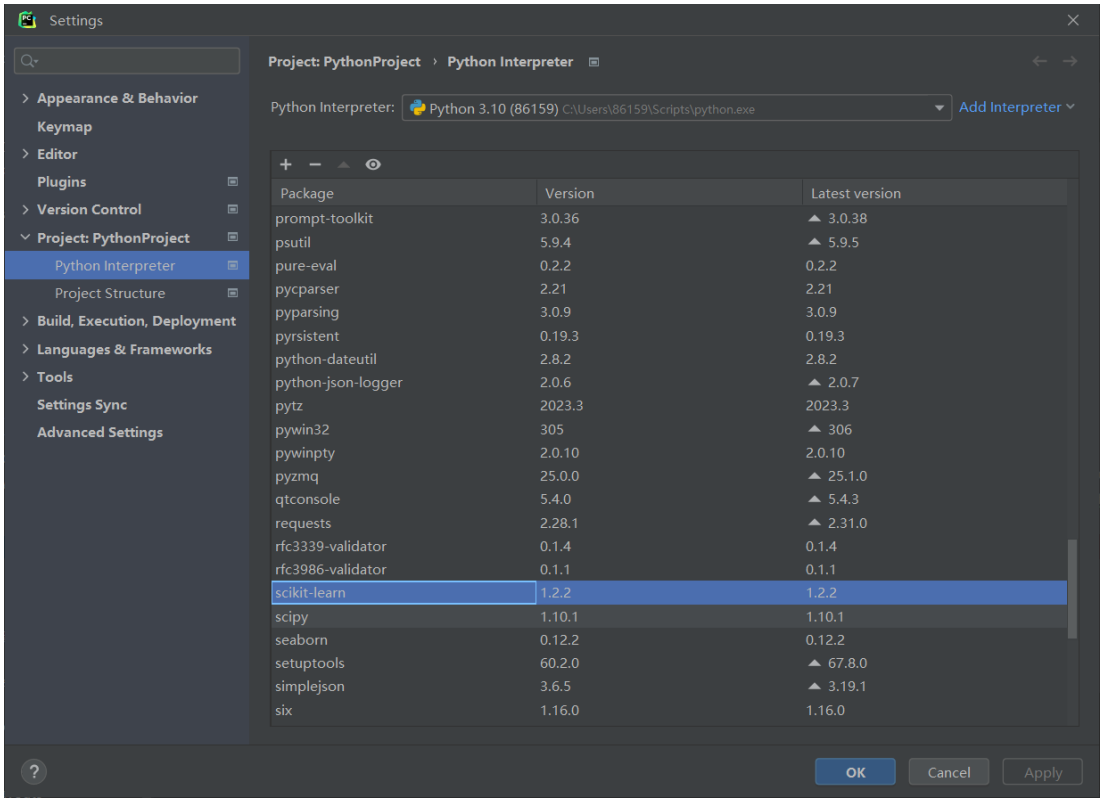
README

1 How to set up the environment

1.1 python编译器的版本以及scikit-learn的版本,确保sklearn的下面有相应的库

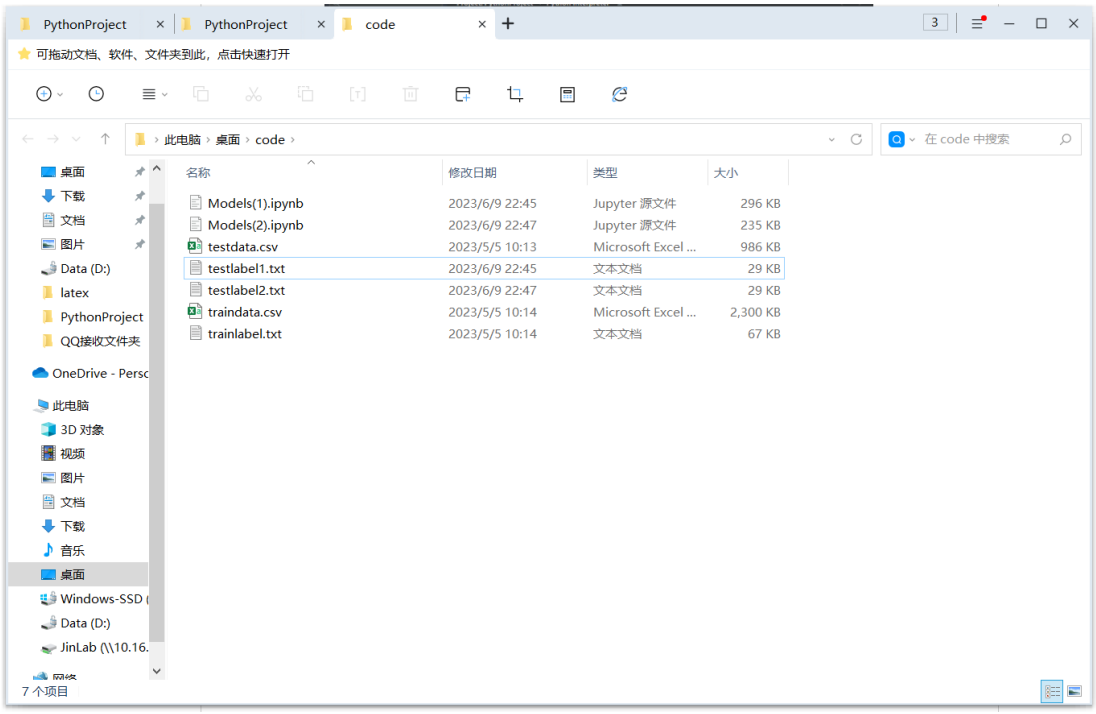


1.2 除了sklearn外还需要导入的相关库，这些库的版本应该问题不大

```
#导入相关库
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2 How to run my code

在code文件夹下面有两个.jupyter文件，这是需要去运行的代码



分别为Model(1)、Model(2)，这两个文件里面涉及数据的导入和数据的预处理–探索数据特征–清洗数据-构建模型–模型预测，模型评估的一系列流程

2.1 之所以这样设计，是因为在两个文件对于数据的预处理的方式不同

Model(1)是采用的将字符型的值转换为整数型的值

Model(2)是采用的独热码编码

2.2 两个文件里面实现的模型

在Model(1)里面实现的是KNN，梯度提升决策树(GBDT)，以及对应的网格搜索的参数优化的KNN和梯度提升决策树，还实现了普通的决策树，三种朴素贝叶斯

在Model(2)里面实现的是Logistic Regression和独热码预处理的梯度提升决策树(GBDT)

两个文件共实现了上面所说的10种模型

2.3 预测结果的存储

Model(1)的预测结果存在testlabel1.txt中

```
#读取testdata，进行预测
df = pd.read_csv(r'.\testdata.csv')
#查看数据缺失情况，
# print(df.isnull().any())
#对无效字符"?"进行处理，在这里我们使用众数替代
for column in df.columns:
    mode=df[column].mode()[0]
    df[column].replace('?',mode,inplace=True)
#删除无用的特征,在原本的居民收入数据集中，关于受教育程度的有两个变量，一个是education（教育水平），另一个是education.num（受教育时长），而且这两个变量的值都是一一对应的，只不过一个是字符型，另一个是对应的数值型，如果将这两个变量都包含在模型中的话，就会产生信息的冗余；fnlwgt变量代表的是一种序号，其对收入水平的高低并没有实际意义。故为了避免冗余信息和无意义变量对模型的影响，考虑将education变量和fnlwgt变量从数据集中删除。
# print(df[['education','education.num']].value_counts())
df=df.drop(['fnlwgt','education'],axis=1)
# 离散型变量的重编码,将字符型的值转换为整数型的值
for feature in df.columns:
    if df[feature].dtype == 'object':
        df[feature] = pd.Categorical(df[feature]).codes
df=df.values
#####################这里转换模型预测#############################
test_label=grid_gbdt.predict(df)
##############################################
path=".\\testlabel1.txt"
np.savetxt(path,test_label,fmt='%d')
```

通过调换提示位置的模型，可以输出不同模型的预测结果到testlabel1.txt中

Model(2)的预测结果存在testlabel2.txt中

```
test_df = pd.read_csv(r'.\testdata.csv')
#离散型变量进行热编码
encode_df = pd.get_dummies(test_df,columns=cat_columns)
#连续型变量归一化
# 对连续型变量进行z-score标准化，经处理数据符合标准正态分布，均值为0，标准差为1,然后，删除未标准化数据，将标准化后数据合并。
num_mean=encode_df[num_columns].mean()
num_std=encode_df[num_columns].std()
num_normal=(encode_df[num_columns]-num_mean)/num_std
encode_df=encode_df.drop(columns=num_columns)
encode_df=pd.concat([encode_df,num_normal],axis=1)

for item in encode_df.columns:
    if item not in features:
        encode_df=encode_df.drop(columns=item)
encode_df=encode_df.values
#################这里调换模型#####################################
test_label=rfc.predict(encode_df)
#####################################################
# test_label
for item in test_label:
    print(item)
```

通过调换提示位置的模型，可以输出不同模型的预测结果到testlabel2.txt中