**作业1：成人死亡率**

姓名：钟洋 学号：22224046

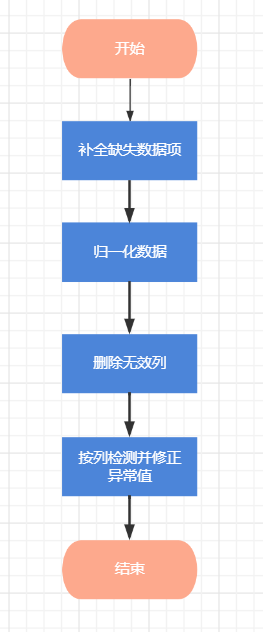
学院（系）专业：航空航天学院电子信息

一、算法描述

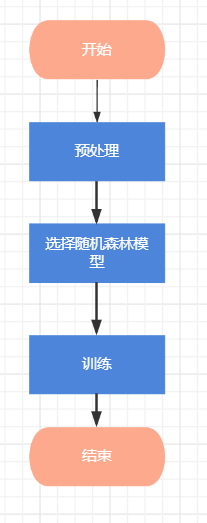
1、算法整体的执行流程

提交的Python文件中包含三个函数，分别是preprocess、model\_fit和predict。整个流程是首先利用preprocess函数预处理训练数据集，接着执行model\_fit函数得到模型，然后提交到平台，平台调用predict函数利用前面训练的模型进行预测，最后评分。

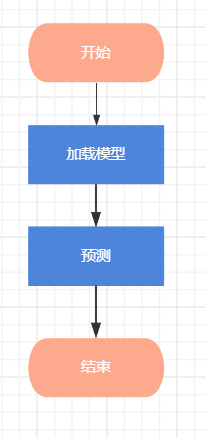
1. 各个函数的执行流程
2. preprocess函数



1. model\_fit函数



1. predict函数



3、源代码

1. preprocess函数

**def** preprocess\_data(data, imputer=**None**, scaler=**None**):

column\_name = ['Year', 'Life expectancy ', 'infant deaths', 'Alcohol',

'percentage expenditure', 'Hepatitis B', 'Measles ', ' BMI ', 'under-five deaths ',

'Polio', 'Total expenditure', 'Diphtheria ', ' HIV/AIDS', 'GDP', 'Population',

' thinness 1-19 years', ' thinness 5-9 years', 'Income composition of resources',

'Schooling']

data = data.drop(["Country", "Status"], axis=1)

**if** imputer==**None**:

imputer = SimpleImputer(strategy='mean', missing\_values=np.nan)

imputer = imputer.fit(data[column\_name])

data[column\_name] = imputer.transform(data[column\_name])

**if** scaler==**None**:

scaler = MinMaxScaler()

scaler = scaler.fit(data)

data\_norm = pd.DataFrame(scaler.transform(data), columns=data.columns)

*# 删除无效列*

data\_norm = data\_norm.drop(['Year', 'Measles ', 'Population'], axis=1)

**for** column\_idx **in** range(data\_norm.shape[1]):

column = np.empty(data\_norm.shape[0], dtype=float)

**for** i **in** range(column.size):

column[i] = data\_norm.iloc[i, column\_idx]

*# 异常值检测*

q75, q25 = np.percentile(column, [75 ,25])

iqr = q75 - q25

min\_val = q25 - (iqr\*1.5)

max\_val = q75 + (iqr\*1.5)

len\_0 = len(np.where(column>max\_val)[0])

len\_1 = len(np.where(column<min\_val)[0])

limit\_0 = len\_0 / len(column)

limit\_1 = len\_1 / len(column)

*# 异常值处理*

new\_column = winsorize(column,(limit\_0, limit\_1))

data\_norm.iloc[:, column\_idx] = new\_column

joblib.dump(imputer, imputer\_filename)

joblib.dump(scaler, scaler\_filename)

**return** data\_norm

1. model\_fit函数

**def** model\_fit(train\_data):

train\_y = train\_data.iloc[:,-1].values

train\_data = train\_data.drop(["Adult Mortality"], axis=1)

train\_data\_norm = preprocess\_data(train\_data)

train\_x = train\_data\_norm.values

*# 线性回归*

*# regressor =linear\_model.LinearRegression()*

*# SVM回归*

*# regressor = svm.SVR()*

*# KNN回归*

*# regressor = neighbors.KNeighborsRegressor()*

*# 随机森林*

regressor = ensemble.RandomForestRegressor(n\_estimators=25)

regressor.fit(train\_x, train\_y)

joblib.dump(regressor, model\_filename)

**return** regressor

1. 算法性能分析
2. 线性回归



1. 支持向量机



1. 最近邻回归



1. 随机森林回归



三、算法进一步研究展望

1、采用不同的数据填充方法，可以分别使用均值、中位数、众数和自定义常量来进行填充并比较算法结果，选择预测最准确的填充方法。

2、调整异常数据的检测与处理方法，比较算法结果，选择预测最准确的结果。