聚丙烯熔融指数数据模型的研究

一，原理分析

1，用到的算法原理

分类，回归，聚类是机器学习的三大问题，而本题目就是典型的回归问题，一共有150组数据，其中每组数据有9个变量与1个因变量。回归本质上是找到能够尽可能准确表示 y 与 X 之间关系的函数$y=f(X)$。机器学习与回归相关的算法有很多，诸如线性回归，多项式回归，支持向量回归，随机森林，K近邻等等。本次大作业主要采用线性回归，多项式回归以及支持向量回归。

线性回归，是在假设$y=\theta_1x_1+\theta_2x_2+\ldots+\theta_nx_n+b$ 的基础上，通过训练集得到最佳斜率和截距的过程。求解方法主要是梯度下降法。

梯度下降法：假设将所有变量看为一个矩阵X，行数为数据量，列数为变量个数。将因变量，参数看作一个向量。想要模型的预测量尽可能离提供的量接近。需要找到一个能够代表拟合程度的量——均方误差，每一次微调中使得在此处参数下的预测量与实际量的均方误差(MSE)越来越小。让均方误差函数对每一个变量求偏导，即得到此次的梯度，梯度的方向则是原函数下降最快的方向，让每一个参数都向梯度的方向走一小步，就使得MSE一点点缩小，使模型拟合程度越来越高。经过矩阵计算得到梯度的表达式为$grad=\frac{1}{m}X^T\cdot(X\cdot\theta-y)$，每次变化后的参数表达式为$\theta_{new}=\theta-\eta\times grad$，其中m指变量的个数，$\eta$指学习率，学习率过大可能导致不能得到较好的模型，学习率过小可能使得迭代次数巨大，效率低。

多项式回归，面对拟合一些曲线的时候，线性回归无论如何也不能得到最佳的模型，这时可以用多项式回归，假设只有一个变量，模型可以表示为$y=\theta_1x+\theta_2x^2+\ldots+\theta_nx^n$，如果有两个及以上的变量，在对每一个变量做这样的变换后，也存在一些交叉项需要考虑，最后的函数十分庞大，不好表示。求解方法也是用梯度下降法，原理类似，只是计算量剧增。

支持向量回归SVR，源自SVM支持向量机，本来是用作分类的，原理是找到一个能够将不同类的数据分隔开的超平面，这个超平面要位于不同类样本的分界处的正中间，间隔最大，鲁棒性最强。如果把超平面表示成$\omega^Tx_i+b=0$，任意点到超平面的距离可表示为$r=\frac{|\omega^Tx+b|}{|\omega|}$，设标签y只能取1或-1。可以规定当$\omega^Tx_i+b>1$时$y_i=1$，$\omega^Tx_i+b<1$时$y_i=-1$，距离超平面最近(即距离为±1)的几个样本成为支持向量，两个异类支持向量的间隔为$r=\frac{2}{|\omega|}$，要找到最大间隔的超平面，即在$\frac{2}{|\omega|}$最大的条件下求ω和b，即$min(\omega,b)\frac{\omega^2}{2}$条件为$y_i(\omega^Tx_i+b\geq1),i=1\ldots m$，是一个条件极值问题。然后再用拉格朗日乘数法加入乘子，再回代得到原问题的对偶问题$max(a)\sum_{i=1}^m a_i-\frac{1}{2}\sum_{i=1}^m\sum_{j=1}^m a_ia_jy_iy_jx_i^Ty_i^T$条件为$\sum_{i=1}^m a_ia_j=0,a_i\geq0$，同时还需满足KKT条件。总之SVM问题的求解只需用到小部分支持向量，重点在于求解向量a，w和b都可由a解得，而a的求解又涉及到SMO算法。在此基础上进行加深，如果在现在有的空间中难以找到超平面，那么先升维再求，即在原问题的基础上添加一个升维的映射。为了避免条件太苛刻而过拟合，又提出了软间隔向量机，允许一部分错误样本。再在此基础上引入SVR，特点为允许y与f(x)存在一定的偏差，$min(\omega,b)\frac{1}{2}|\omega|^2+C\sum_{i=1}^m l_\epsilon(f(x_i)-y_i)$，其中$l_\epsilon$是$\epsilon$-不敏感损失函数通过引入松弛变量，拉格朗日乘数法等等一系列复杂数学计算，又可得到一个对偶问题，解出对偶问题的向量a即可求解出超平面。除此之外还有核函数方法等等求解方法。总之SVR是可以在一个更高维度的特征空间找出一个超平面覆盖大部分数据的方法，其精度应该比低维度拟合更好。

评估模型性能的指标：均方误差MSE：预测值与真实值之间差异的平方的平均值，MSE越小，精度越高,tong；决定系数R^2，介于0和1之间，用于衡量模型对因变量变异的解释程度，越接近1证明拟合效果越好。

所用编程语言python，编译器jupyter notebook

2，问题背景及意义

本作业的大背景是找到测量熔融指数的方法，熔融指数决定了生产产品的牌号与价格，但是难以直接测量，这个背景有着非常现实的需求，找到此模型对应提高生产质量提高收益有着非常重要的意义。本次作业是用九个可直接观测的变量经过建模得到的。用人工智能的方法不仅可以得到由这九个参数得到目标指数的模型，也可以了解哪些参数对结果的影响最大。以此为代码核心，连接好传感器，设计好前后端就可以直接包装成软件，就是物联网的一个典型例子。除此之外，与之类似的各种案例，都可以用相同的方法来解决，本质上都是数据的回归问题。

二，代码实现

1，数据处理

数据是一个.xls表格，包含150组数据。首先用pandas库中的read_excel函数读取表格，再分别取其不同列的数据划分得到变量矩阵X和因变量向量Y。

[179]:
```
import pandas as pd
import numpy as np
data_frame=pd.read_excel("data150（人工智能数值类）.xls")
data_array=data_frame.values
X_data=data_array[:,1:10]
Y_data=data_array[:,11]
print(X_data.shape)
print(Y_data.shape)
```

```
(150, 9)
(150,)
```

用sklearn库自带的train_test_split函数随机划分训练集和测试集，其中训练集和测试集的比例为7:3。

[180]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X_data, Y_data, test_size=0.3, random_state=42)
print(X_train[0])
print(X_test[0])
```

```
[-1.38175011  1.1889465  -0.19668864 -0.42327314  0.46867149  0.62135256
 -1.22675225  0.32989877  0.57520302]
[-0.65166227 -0.38266778 -0.17767878  0.44665239  0.4260908   0.62273461
 -1.3042393  -0.66506817  0.08242785]
```

2，多元线性回归

首先用线性回归拟合模型，调用sklearn库中的LinearRegression函数。

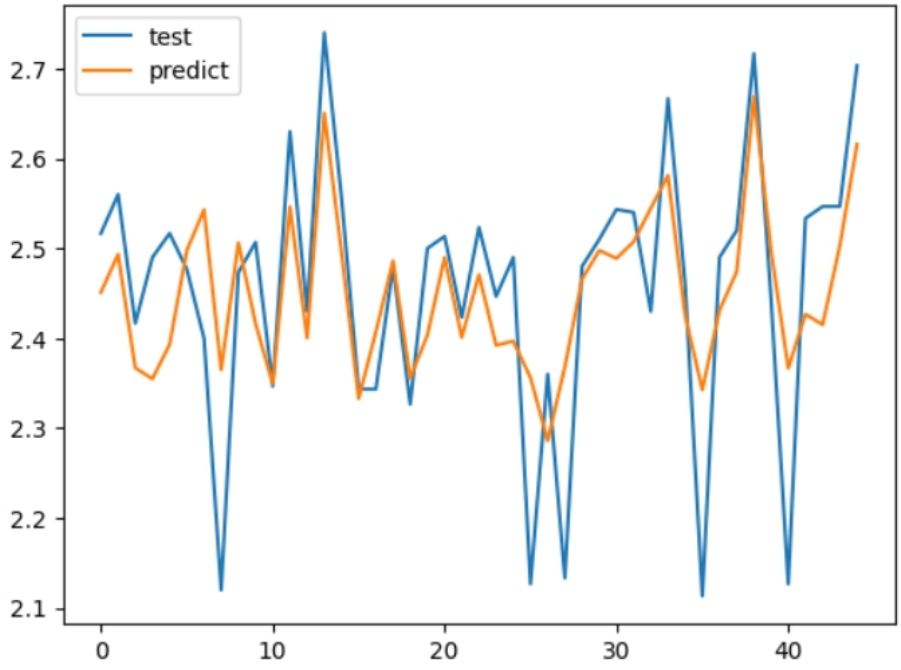
[181]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train, Y_train)
Y_pred = model.predict(X_test)
print(Y_pred)
```

```
[2.45109696 2.49342219 2.36697678 2.35487113 2.39324458 2.49774143
 2.54296776 2.36517373 2.50618377 2.41639073 2.34910362 2.54620424
 2.40044668 2.65044855 2.49607096 2.33291559 2.40686273 2.48617412
 2.35546203 2.40353764 2.4897997  2.40102896 2.47067376 2.39218701
 2.39680678 2.3560713  2.28604001 2.36809832 2.46567458 2.49767168
 2.48888226 2.50723687 2.54458541 2.58095234 2.42522086 2.34258197
 2.43113859 2.47405574 2.66869944 2.49781119 2.36663602 2.4267034
 2.41510872 2.50263198 2.61570044]
```

为了评估模型性能，调用sklearn库中的mean_squared_error和r2_score函数来得到模型的MSE和R方。同时调用matplotlib库使数据可视化。

[214]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error, r2_score
print("MSE:",mean_squared_error(Y_test, Y_pred))
print("R^2:",r2_score(Y_test, Y_pred))
X=range(0,45)
plt.plot(X,Y_test, label='test')
plt.plot(X, Y_pred, label='predict')
plt.legend()
plt.show()
```

```
MSE: 0.010730598480313124
R^2: 0.5154259824934877
```



模型的斜率与截距如下：

[211]:
```
X_k=model.coef_#得到模型斜率
X_b=model.intercept_#得到模型截距
print(X_k)
print(X_b)
```

```
[-0.02039217 -0.02380072 -0.03199078  0.0467218  -0.01859704  0.00930802
  0.02861808  0.00886494  0.01351688]
2.446381950362378
```

发现线性回归R方仅有0.515，精度太低。

3，梯度下降法

用梯度下降法的思路求一遍，看结果是否与用线性回归函数的结果相等。

为了把截距也包括进去可以将斜率与截距合并为一个向量，并且在每个数据的最后面添加一个固定的1，这样变化后的$X \cdot \theta$就直接是预测值了.

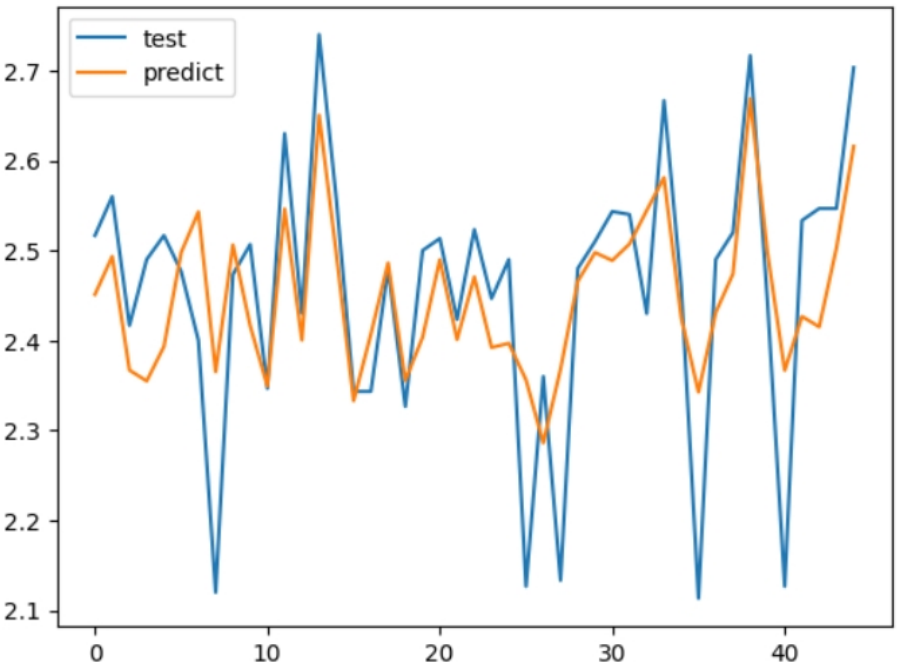
```
X_train_added=np.append(X_train,np.ones((X_train.shape[0],1)),axis=1)#在训练集每组数据末尾添加一个1，这样乘出来结果对应正好是加截距
X_test_added=np.append(X_test,np.ones((X_test.shape[0],1)),axis=1)
```

设原始的参数列向量全为0，学习率为0.01，迭代次数为1000，m=10，由梯度下降的公式有：

```
n=0.01
iteration=1000
m=10
for i in range(iteration):
    error=np.dot(X_train_added,theta_1)-Y_train
    grad=(1/m)*np.dot(X_train_added.T,error)
    theta_1=theta_1-n*grad
print(theta_1)
Y_pred_grad_1=np.dot(X_test_added,theta)
```

最后的结果与线性回归完全相等

```
[-0.02045179 -0.02379773 -0.03196928  0.04672476 -0.01858323  0.00929544
  0.02863661  0.00893825  0.01346576  2.44638113]
```




```
MSE: 0.010730598480313124
R^2: 0.5154259824934877
```

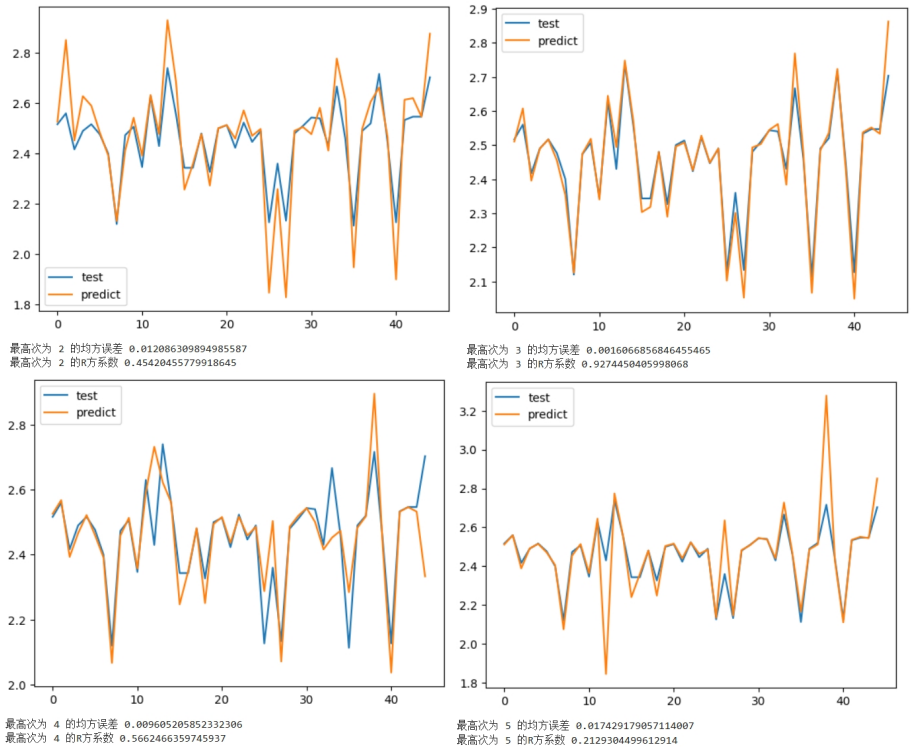
4，多元多项式回归

发现用线性回归得不到精度较高的模型，说明原数据直接并不存在较为明显的线性关系，遂用多项式进行拟合

多项式拟合要用到PolynomialFeatures函数用于生成多项式，make_pipeline用于将线性回归模型与多项式连接起来构成多项式回归模型。由于不确定最高项数取多少，于是搞个循环：

```
•[241]: from sklearn.preprocessing import PolynomialFeatures
        from sklearn.pipeline import make_pipeline
        for i in range(2,6):
            poly = PolynomialFeatures(degree=i)
            line=LinearRegression()
            model_2=make_pipeline(poly, line)
            model_2.fit(X_train, Y_train)
            Y_pred_poly=model_2.predict(X_test)
            plt.plot(X,Y_test, label='test')
            plt.plot(X,Y_pred_poly, label='predict')
            plt.legend()
            plt.show()
            print("最高次为",i,"的均方误差",mean_squared_error(Y_test, Y_pred_poly))
            print("最高次为",i,"的R方系数",r2_score(Y_test, Y_pred_poly))
```

得到的结果为：



可以得到最高次数为3时的拟合精度最好，其MSE=0.0016，R方=0.9274。

5，支持向量回归SVR

进行支持向量回归前先要对数据进行归一化处理

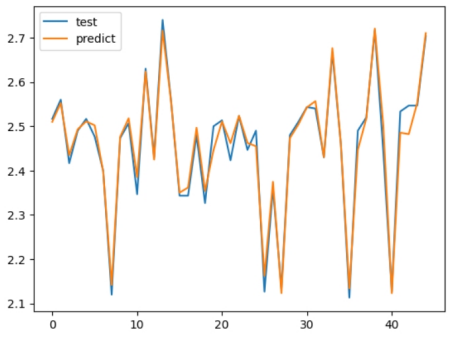
```
scaler = StandardScaler()#数据归一化
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

然后调用sklearn库中的SVR函数

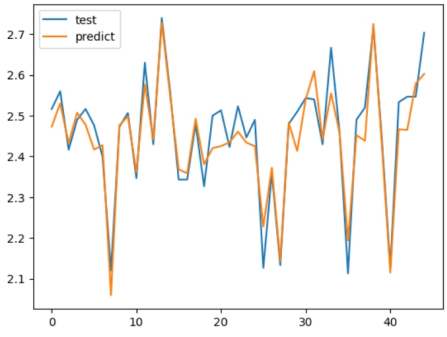
```
for i in ['rbf','sigmoid','poly','linear']:
    model_3 = SVR(kernel=i, C=1.0, epsilon=0.001)
    model_3.fit(X_train_scaled, Y_train)
    Y_pred_svr = model_3.predict(X_test_scaled)
    plt.plot(X,Y_test, label='test')
    plt.plot(X,Y_pred_svr, label='predict')
    plt.legend()
    plt.show()
    print("所用核函数为：",i)
    print("MSE:",mean_squared_error(Y_test, Y_pred_svr))
    print("R^2:",r2_score(Y_test, Y_pred_svr))
```

其中SVR函数的三个参数分别代表：kernel核函数，C对错误分类的惩罚力度，epsilon对数据拟合的宽松程度。其中C过大会导致过拟合，C过小会导致欠拟合；epsilon越小，拟合精度越高。

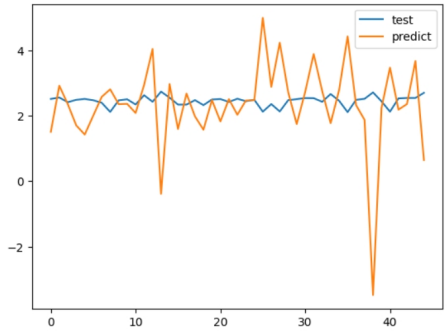
由rbf，sigmoid，poly，linear四个核函数得到结果有：



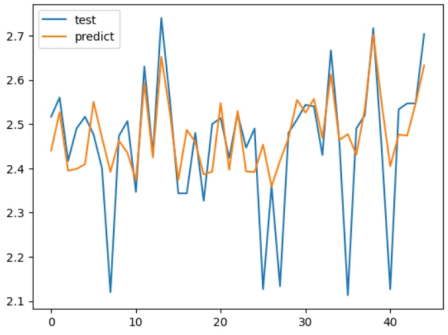
所用核函数为： rbf
MSE: 0.0005647575231538242
R^2: 0.9744965928588458



所用核函数为： poly
MSE: 0.0026064339965357875
R^2: 0.8822982524801314



所用核函数为： sigmoid
MSE: 1.928650432555574
R^2: -86.09429303349125



所用核函数为： linear
MSE: 0.013578828572868804
R^2: 0.38680516965953504

发现用rbf核函数的模型拟合精度最高，MSE仅有0.000564，R方达到了0.9745。

5，最终结果

发现用支持向量回归得到的模型精度最高。R方=0.9745