软件技术基础大作业——聚丙烯熔融指数预测

一、 问题背景

1. 聚丙烯熔融指数预测

数据来源：聚丙烯生产过程熔融指数数据，共 150 组时序数据，见 excel 表格。问题描述：聚丙烯生产过程中的熔融指数是一个重要质量控制指标（excel 表格中的变量 y），决定了所生产产品的牌号与价格，但是难以在线测量，从而导致生产质量的控制品质大大降低。因此，采用生产中与该质量控制指标相关的可直接测量的操作变量（excel 表格中的变量 x，共 9 个），来在线预测该质量控制指标，从而提高生产质量控制品质。

二、 实验任务

根据所给的数据集，建立一个模型，从而能够预测聚丙烯熔指指数指标。

三、 设计过程

初步拿到数据，发现有 9 个输入变量，1 个输出变量。题目要求是希望能够在线预测质量控制指标，从而提高生产质量控制品质。所以初步判断这是一个多元的回归问题。根据所学到的知识以及所查阅的资料，我首先的想法是通过使用 sklearn 进行数据集训练与模型建立。同时利用 pandas 库和 numpy 库对数据进行操作。

1. 数据处理

题目所提供数据为 xls 格式，如下图所示：

序号	x1	x2	x3	x4	x5	x6	x7	x8	x9		y
1	2.18626	-0.98362	-0.48902	2.655448	-2.81004	0.602695	1.307641	2.701685	0.172023		2.65
2	2.304016	-0.97855	-0.49578	2.628263	-3.05133	0.607532	1.320871	2.62639	0.351214		2.673333
3	2.421772	-0.97347	-0.50254	2.601077	-3.29262	0.612369	1.3341	2.551095	0.530405		2.696667
4	2.539528	-0.9684	-0.5093	2.573892	-3.53391	0.617206	1.34733	2.4758	0.709596		2.72
5	2.127382	-1.11048	-0.50254	2.336022	-3.43456	0.615824	1.35111	1.55075	-0.58954		2.716667
6	1.715235	-1.25256	-0.49578	2.098152	-3.3352	0.614442	1.354889	0.6257	-1.88867		2.713333
7	1.303089	-1.39465	-0.48902	1.860281	-3.23585	0.61306	1.358669	-0.29935	-3.18781		2.71
8	1.108791	-1.3497	-0.47564	1.860281	-2.98036	0.61306	1.33599	0.082502	-2.33665		2.72
9	0.914494	-1.30476	-0.46226	1.860281	-2.72488	0.61306	1.313311	0.464354	-1.48549		2.73
10	0.720196	-1.25981	-0.44889	1.860281	-2.4694	0.61306	1.290632	0.846206	-0.63434		2.74
11	0.84384	-1.27359	-0.44621	1.139874	-2.34165	0.610296	1.264173	0.926879	-0.36555		2.703333
12	0.967484	-1.28736	-0.44354	0.419467	-2.21391	0.607532	1.237714	1.007552	-0.09676		2.666667
13	1.091128	-1.30113	-0.44086	-0.30094	-2.08617	0.604768	1.211255	1.088225	0.172023		2.63
14	1.026362	-1.32433	-0.43889	-0.53201	-1.83069	0.604768	1.211255	1.10436	0.216821		2.6
15	0.961596	-1.34753	-0.43692	-0.76309	-1.5752	0.604768	1.211255	1.120494	0.261619		2.57
16	0.89683	-1.37073	-0.43495	-0.99416	-1.31972	0.604768	1.211255	1.136629	0.306417		2.54
17	0.802626	-1.3468	-0.44818	-0.94659	-0.93649	0.606841	1.164007	1.120494	0.306417		2.546667
18	0.708421	-1.32288	-0.46142	-0.89901	-0.55327	0.608914	1.116758	1.10436	0.306417		2.553333
19	0.614216	-1.29896	-0.47465	-0.85144	-0.17004	0.610987	1.06951	1.088225	0.306417		2.56
20	0.649543	-1.08149	-0.46353	-0.77668	-0.01391	0.610987	1.060061	1.373269	0.754394		2.556667
21	0.684869	-0.86401	-0.45241	-0.70192	0.14222	0.610987	1.050611	1.658314	1.202371		2.553333
22	0.720196	-0.64654	-0.44128	-0.62716	0.298349	0.610987	1.041161	1.943359	1.650349		2.55
23	0.873279	-0.54432	-0.43297	-0.64075	0.411897	0.612369	1.031712	1.803525	1.471158		2.543333
24	1.026362	-0.44211	-0.42467	-0.65435	0.525446	0.613751	1.022262	1.663692	1.291967		2.536667
25	1.179445	-0.3399	-0.41636	-0.66794	0.638994	0.615133	1.012812	1.523859	1.112776		2.53
26	1.150006	-0.41311	-0.41284	-0.45046	0.553833	0.613751	0.995803	1.136629	0.620001		2.526667

图 1: 原始数据

为了方便 python 操作，将数据第一列删除，'y' 列与左边靠拢，并保存为 csv 格式。

调用 corr 函数，查看各个输入变量 x 与 y 之间的相关系数，并通过 seaborn 库绘制可视化相关程度的热图如下：

图 2: 输入输出相关性

可以看出输入变量 x1-x9 与输出变量 y 的相关性普遍比较低，最高的也只是 x3，x4，x5，x6 这几个变量。

于是我决定先选用 x3，x4，x5，x6 这四个输入作为训练输入，由于无法确定该问题是线性回归还是非线性回归，于是我首先尝试了线性回归。

先利用 train_test_split 函数将数据集划分为训练集和测试集

```
X_train2, X_test2, Y_train2, Y_test2 = train_test_split(X_Poly, y, train_size=0.80,
    random_state=1)
```

这里将 150 个数据随机分成 120 个训练集 X_train1，Y_train1 和 30 个测试集 X_test1，Y_test1。

随后利用 scikit-learn 框架中的 LinearRegression 模型中，使用 fit 函数进行训练，最后得到对应的线性回归方程。得到模型结剧 a，回归系数 b。

```
model = LinearRegression()
model.fit(X_train, Y_train)
print("截距: ",LinearR.intercept_)
print("系数: ",LinearR.coef_)
```

得到结果如下：

```
截距：2.4513199017988554
系数：[-0.0391533 -0.03018913 -0.03880207 0.05311798 -0.00549697 0.01419835
0.05236998 0.01524657 0.0052771 ]
```

```
y = -0.0391533*x1 + -0.03018913*x2 + -0.03880207*x3 + 0.05311798*x4 + -0.00549697*x5 +
    0.01419835*x6 + 0.05236998*x7 + 0.01524657*x8 + 0.0052771*x9
```

在得到结果后，通过模型的 predict 函数对数据进行了预测，并且使用模型的 mean_squared_error 函数和 R2 检测 r2_score 函数进行了模型评分。

```
Y_pred1 = LinearR.predict(X_test1)
# 平均误差相对于样本真实值均值偏差
print("mse相对于测试值平均值的偏差：", np.sqrt(MSE(Y_test1, Y_pred1))/Y_test1.mean())
# R^2检测
print("r2检测：", r2_score(y_true=Y_test1, y_pred=Y_pred1))
```

得到结果如下：MSE 相对于测试值平均值的偏差：0.03954403176473497

R^2 检测结果为：0.2770634859513812

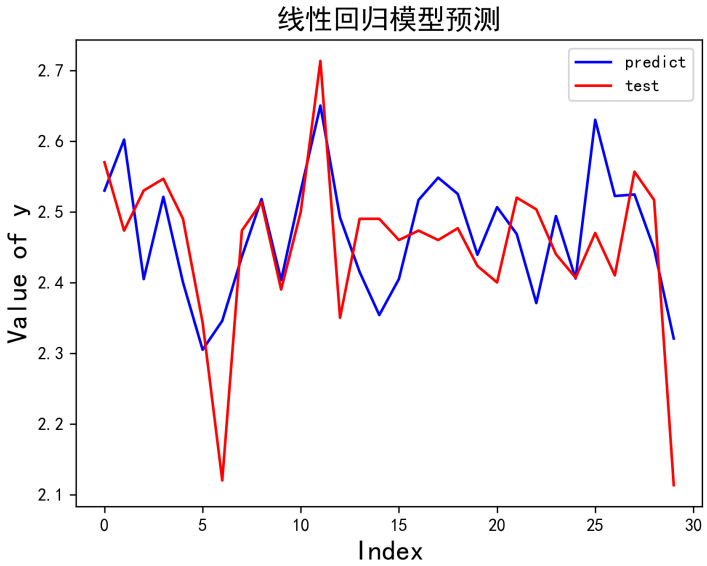

图 3: 预测和测试输出比较

从结果可以看出，训练得到的线性模型性能并不长非常的理想。于是我又尝试用非线性回归模型进行预测。此处我先尝试了用二次多项式回归模型。

首先需要给原来的一次特征 x1 x9 添加二次特征，此处用到了 sklearn 库里的 PolynomialFeatures 函数。并利用 fit_transfrom 对特征进行预处理（标准化、归一化等）。

```
PolyF = PolynomialFeatures(degree=2, interaction_only=False, include_bias=False)
X_Poly = PolyF.fit_transform(x)
```

然后同样对数据集进行分割，并使用 LinearRegression 模型进行训练，得到结果如下：

```
X_train2, X_test2, Y_train2, Y_test2 = train_test_split(
    X_Poly, y, train_size=0.80, random_state=1)
LinearR2 = LinearRegression().fit(X_train2, Y_train2)
print("截距：",LinearR2.intercept_)
print("系数：",LinearR2.coef_)
```